MODELING NATURAL-IMAGE SPACES FOR SINGLE-LABEL IMAGE CLASSIFICATION & PHOTO-REALISTIC STYLE TRANSFER AND DIRECTIONALLY PAIRED PRINCIPAL COMPONENT ANALYSIS FOR THE ESTIMATION OF COUPLED DATA

A Dissertation
Presented to
The Academic Faculty

By

Yifei Fan

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

December 2020

MODELING NATURAL-IMAGE SPACES FOR SINGLE-LABEL IMAGE CLASSIFICATION & PHOTO-REALISTIC STYLE TRANSFER AND DIRECTIONALLY PAIRED PRINCIPAL COMPONENT ANALYSIS FOR THE ESTIMATION OF COUPLED DATA

Approved by:

Dr. Anthony Yezzi, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Mark Davenport
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Patricio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Shijie Deng
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Frieder Ganz
Digital Imaging
Adobe Inc.

Date Approved: August 27, 2020

When you have eliminated the impossible, whatever remains, however improbable, must be the truth.

Sir Arthur Conan Doyle, stated by Sherlock Holmes

Dedicated to my parents.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to Dr. Anthony Yezzi, who taught me how to conduct research, and more importantly, to be a better person. I would also like to extend my deepest gratitude to my committee Dr. Mark Davenport, Dr. Patricio Vela, Dr. Shijie Deng, and Dr. Frieder Ganz, for generously offering their time and support throughout the reviewing process. I'm extremely grateful to the team where I spent two most memorable summers as an intern at Adobe DI. I'm deeply indebted to my intern manager Mark Nichoson, who offered me a research topic and a chance for a career at the one-of-a-kind community. Several mentors and collaborators have played a decisive role in my exceptional experience at Adobe, including Frieder Ganz, Peter Merrill, Gregg Wilensky, Olena Soroka, and Simen Chen.

I started graduate school as one of the first students at Georgia Tech Shenzhen campus. Especially helpful to me during my master's program were director Dr. G. Tong Zhou and program manager Zhiya Wang, who offered valuable advice on career development and application for Ph.D. programs. I also wish to thank Dr. Chuanyi Ji, who taught me the first course on machine learning in Shenzhen and sparked my interest in the research domain. In 2016, I transferred to our Ph.D. program and moved to the Atlanta campus, where I spent four years studying and working with the most talented and diligent professors and students. I cannot leave Georgia Tech without mentioning the two most inspiring courses that have motivated my research: CS7476 Advanced Computer Vision and ECE6560 PDEs in Image Processing and Computer Vision. Many thanks to Dr. James Hays and Dr. Anthony Yezzi for providing such a wonderful learning experience. Besides the relentless support for studying knowledge in my research domain, I also gratefully acknowledge the academic writing assistance from Jane Chisholm and Dr. Lauren Lukkarila.

The completion of my study would not have been possible without the following financial supports. I very much appreciate Dr. Zhaohua Wang for the job as a graduate research

assistant for two years. I also enjoyed working with Dr. Hongyuan Zha and Dr. Henry Owen as a graduate teaching assistant. Thanks should also go to Dr. Avinash Ravichandran, who kindly offered me an internship at Amazon. I would also like to extend my gratitude to colleagues in the Laboratory of Computational Computer Vision as well as collaborators outside of Georgia Tech, including Navdeep Dahiya, Minas Benyamin, Chao Tang, Samuel Bignardi, Alper Yildirim, Robert Friedlander, Huizong Yang, Honglin Li (University of Surrey) and Dr. Payam Barnaghi (University of Surrey). Thank you all for bringing insightful suggestions to our discussions and useful contributions to the collaborative work.

Last but not least, I am deeply grateful to my parents, who always nurtured and supported me throughout my life. I sincerely thank my beloved angel, Lyric, for the unwavering love and encouragement, as well as the assistance in preparing the complex graphics for this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Despite the tremendous success of machine learning and data-driven approaches in the past decade, machine perception is much less robust and explainable compared to human perception. In this dissertation, we aim at a methodology that provides better explainability of machine-learning algorithms both under the linear-analysis framework and in the deep-learning domain. Under the linear-analysis framework, Principal Component Analysis (PCA) is a classic approach that offers better explainability via dimension reduction. In this dissertation, we extend the classic PCA for coupled yet partially observable test data. The proposed Directionally Paired Principal Component Analysis (DP-PCA) is the optimal linear model that performs dimension reduction and least-square regression between the coupled variable sets in the principal subspace, which leads to the lowest estimation errors at a fast speed and with the least storage requirement.

In the deep-learning domain, we provide a unified explanation of the behaviors of various machine-learning algorithms and the gap between human and machine perception with the proposed conceptual model of natural-image spaces. By formulating classification as a partition of the image space, we further provide a topological view of knowledge in machine perception by defining fundamental concepts, including information, knowledge, beliefs, and truths. We illustrate the benefits of the proposed conceptual image-space model and the topological view of knowledge in two concrete applications: single-label classification and photo stylization.

- For single-label classification, we demonstrate the usage of two types of hidden information in the image space, "poorly justified true beliefs" and "false beliefs," on im-

proving and preserving classification accuracy. The usage of adversarial examples in the hidden information implies that the discrepancy between benign and adversarial examples are not irreconcilable. Through an empirical study on test-time smoothing defense against adversarial attacks, we present a non-monotonic relation between attacks and defenses as well as the large variance in robustness among categories and samples. Following the properties of natural-image spaces, we verify geometric causes of adversarial examples through carefully-designed controlled experiments.

- For photo style recognition and transfer, we re-partition the image space with artistic presets and establish a controlled photo style recognition benchmark which disentangles styles from contents. Consequently, the style classifier behaves very differently from a content classifier in various vision tasks, and it can provide useful guidance signals which support global style transfer. By modeling the image space at the object level, we ensemble a content-aware local style transfer pipeline in which the proposed segmentation refinement module removes defects from inaccurate segmentation maps and supports feature blending at various levels.

CHAPTER 1
INTRODUCTION

We live in an age full of data, but what is lacking is the understanding and interpretability of the data. Researchers often believe that the dimension of useful features in data is generally much lower than that of the data themselves, which implies that plenty of redundancy or irrelevant information exists in data samples. Such redundancy or irrelevance in data samples leads to unnecessary complexity and the "curse of dimensionality" [1]. To reduce the complexity in data and provide better explainability with useful features, various dimension-reduction approaches have been proposed, including Principal Component Analysis (PCA) [2]. Those classic linear-analysis frameworks, however, become less capable and scalable in the deep-learning era, especially when both the dimension and number of data samples increase dramatically.

The past decade has witnessed the success of artificial intelligence (AI) and machine learning in various computer-vision tasks, from fundamental problems such as classification [3] and detection [4] to artistic applications, including image generation [5, 6] and style transfer [7]. One of the most crucial components in modern AI is artificial neural networks. Although inspired by biological neural networks, artificial neural networks perceive information very differently from our brains. There is a vast gap between human perception and machine perception, which leads to uncertainties and concerns on the robustness of machine-learning algorithms. In general, at least three significant differences exist between modern AI and our brain. One difference lies in the ability of generalization: humans can learn from only a few samples and generalize effectively by analogy, whereas AI suffers from domain gaps or dataset bias [8]. Secondly, despite visual illusions, human perception is much more robust under signal disturbance. On the contrary, neural networks can be misled by adversarial examples [9], even though the perturbations are almost im-

perceptible to humans. The third difference is that humans excel at preserving previously-learned knowledge in a continual-learning environment. Unfortunately, for AI systems, catastrophic forgetting [10] is one of the biggest obstacles when it comes to lifelong learning: a classifier forgets categories that learned from early tasks once the categories are no longer revisited during training. Those uncertainties in machine learning prevent us from building robust and interpretable real-world applications for recognition. Similar situations occur in the domain of artistic applications, as algorithms on image generation and style transfer are subject to the risk of producing unpredictable and baffling outcomes.

Previous studies have separately addressed the above differences and issues primarily from a statistical perspective by regarding images as samples from various distributions. As test datasets sometimes fall out of the distribution of training samples, generalization may fail, and bias could occur. Similarly, adversarial examples are considered out of the distribution of benign samples that belong to the image manifold. Although previous studies have managed to preserve and utilize knowledge learned from one domain (i.e., training set or source distribution) for solving tasks in another (i.e., test set or target distribution), they have not provided an answer for a simple yet fundamental question: *what exactly is knowledge in machine perception*, or equivalently *what exactly has been learned by a neural network after training?* We believe it necessary to formally address the question as it may reveal important characteristics of machine perception, which differs substantially from human perception. A better understanding of those fundamental questions can open possibilities for novel designs and applications when tackling existing problems.

1.1 Research Objectives

The objective at the outset of this research was (and still is) to study and model the space of natural images from an AI perspective so that we can better explain the behaviors and characteristics of machine perception. The understanding will serve as principles to guide machine-learning algorithms in solving computer-vision tasks, including single-label im-

age classification and photo-realistic style transfer. The motivation of this research is that the prevailing "image-manifold" model cannot sufficiently explain the uncertainties in machine learning, and a *unified* understanding of the robustness issues in machine-learning algorithms is missing. To address the issues, we interpret the fundamental classification task as a partition of an input space, in which context we study the properties of the input spaces and further define basic concepts such as *information*, *knowledge*, *beliefs*, and *truths*. The research then aims to explain and mitigate the uncertainty issues such as dataset bias and adversarial examples in single-label classification tasks with thoughtful strategies that account for the geometry and topology of the natural-image spaces. In the artistic domain, images spaces are required to be re-partitioned in a controlled manner that disentangles photo styles from contents for better comprehension. Moreover, modeling the image spaces at multi-scales is necessary to support content-aware object-level local style transfer. As the research evolved and matured, the objective broadened after noticing the loss of optimal estimation in classic cross-decomposition methods for coupled datasets. Accordingly, we establish an additional objective for this research: extending the classic linear dimension-reduction and regression analysis framework such that it provides optimal estimation for coupled and partially observable data.

1.2 Key Contributions

In summary, the major contribution of this work is fourfold:

- We propose a conceptual model of natural-image spaces and a topological view of knowledge for machine perception. With the variational-calculus view of machine learning and the defined fundamental concepts of AI-knowledge, we provide a unified explanation of various studies in machine learning. By formulating classification as a partition of the image space and examining the properties of the space, we elaborate the gap between human and machine perception and reveal hidden information in the image space that used to be overlooked in previous studies.

- We verify the hypothetical image-space model in the task of single-label classification and demonstrate the potential benefits from adopting the philosophy from the proposed conceptual model and topological view of knowledge. In specific, we illustrate the usage of two hidden information, "poorly justified true beliefs" and "false beliefs," on improving and preserving classification accuracy. The usage of adversarial examples in those two hidden information implies that the discrepancy between benign and adversarial examples are not irreconcilable. Through an empirical study on defending adversarial robustness with smoothing techniques, we present the non-monotonic relation between attacks and defenses as well as the large variance in robustness among categories and samples. Finally, we verify several causes of adversarial examples, including path-connected regions, an excessive number of target categories, and the geometry of the categories.

- We study photographic style recognition and photo-realistic style transfer under the guidance of the proposed image-space model. The equivalence of various color spaces is verified in space translation tasks. By re-partitioning the image space with artistic presets, we established a controlled photographic style recognition benchmark which disentangles styles from contents. We then exhibit the differences between content and style classifiers via comparison on various vision tasks and illustrate the beneficiary guidance from style recognition in global style transfer. For content-aware local style transfer, we ensemble a fully automatic pipeline in which the proposed segmentation refinement module removes defects form inaccurate segmentation maps and supports feature blending at various levels.

- We propose Directionally Paired Principal Component Analysis, the optimal linear method for estimating coupled data in cases that part of the variables becomes unobservable at test time. The method is capable of estimating the unobservable variables either unconditionally or under the condition that the variability of the observable

part is optimally expressed. Through data reconstruction and prediction experiments on synthetic and real datasets, we prove that compared with existing approaches, the proposed Directionally Paired PCA achieves the lowest estimation errors and at a faster speed.

1.3 Organization of the Dissertation

The rest of the dissertation is organized as follows. Chapter 2 reviews all related studies from various fields in literature. Chapter 3 introduces the proposed hypothetical image-space model and the topological view of AI knowledge. Chapters 4 and 5 illustrate applications of the proposed image-space model in single-label image classification and photo-realistic style transfer. Chapter 6 presents a collaborative work on Directionally Paired PCA, which performs dimension reduction and regression analysis to estimate coupled yet partially observable data. Finally, Chapter 7 concludes the dissertation with summaries and discussions.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we review relevant studies in the literature. We begin in Section 2.1 by filling the blank in the definition of image manifold with the one from mathematics. Sections 2.2 and 2.3 review studies on dataset bias and adversarial examples, the two robustness issues in single-label classification, respectively. In Section 2.4, we survey previous research on knowledge representation with neural networks. Finally, applications in image style transfer is reviewed in Section 2.5.

2.1 Natural Image Manifold

It is often believed that natural images are embedded in a manifold whose dimension is much lower than that of the images themselves. However, the assumption remains at a conceptual level, and a formal definition of the image manifold is missing. In mathematics, a topological space $(\mathcal{M}, \mathcal{O})$ is called a d-dimensional manifold if $\forall p \in M : \exists\, p \in \mathcal{U} \in \mathcal{O}$ with a continuous map $x : \mathcal{U} \to x(\mathcal{U}) \subseteq \mathbb{R}^d$ such that 1): x is invertible and 2) x^{-1} is continuous [11]. The pair $(\mathcal{U}, x)$ of the open set and mapping is called a chart. A collection of charts $\mathcal{A} = \{(\mathcal{U}_{(\alpha)}, x_\alpha) \mid \alpha \in A\}$, in which the label α can come from any arbitrary index set A, is called an atlas of the manifold if $\bigcup_{\alpha \in A} \mathcal{U}_{(\alpha)} = M$, meaning that the entire manifold can be covered by all charts. Manifolds have important topological properties and a manifold is connected if and only if it is path connected [12]. Intuitively, the space locally resembles real d-dimensional Euclidean space.

In computer vision, researchers manage to approximate an open set $\mathcal{U}$ of the manifold and the mapping x via learning [13]. Before the deep-learning era, mappings x from the image space to target (e.g., label) spaces have been studied for solving tasks from classification to segmentation. With the recent advance in computation, estimating the inverse

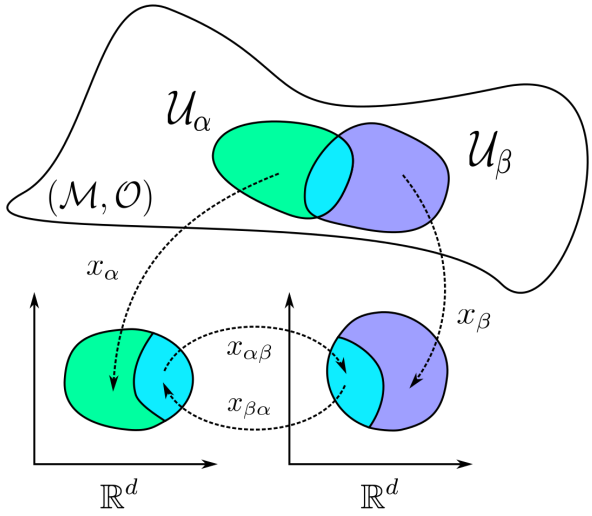


Figure 2.1: Two charts $(\mathcal{U}_\alpha, x)$, $(\mathcal{U}_\beta, x)$ on a manifold $(\mathcal{M}, \mathcal{O})$

mappings x^{-1}, or even traversing the image manifold [14] becomes possible. Generative adversarial networks (GANs) [15], which will be reviewed at Chapter 2.5.4, are state-of-the-art techniques for the above tasks. Despite the advantages of a simple and elegant mathematical form, the manifold model does not fully account for that spaces of images are vast and sparsely populated [16]. In practice, a mapping learned from training samples in one chart may fail on another chart.

2.2 Dataset Bias

The issue of dataset bias was formally pointed out in [8], after the authors became aware that researchers who had worked in object and scene recognition could easily guess which images came from which datasets. They claimed that such a difference among datasets could even be captured by a classifier and reflected in the diagonal pattern of the confusion matrix. Although the goal of computer-vision datasets is to offer unbiased representations of the world, they seem to express them with a strong built-in bias. A follow-up study [17] exploited dataset bias during the training phase. The authors proposed two sets of weights, bias vectors and visual world weights, to undo the damage of dataset bias. They demon-

strated the benefit of explicitly accounting for bias when multiple datasets are involved. Another study [18] provided a large-scale analysis of the issue with twelve databases. However, as most studies have addressed the issue at the dataset level, they do not help explain or eliminate bias in practice. It is possible that some data samples we collect resemble one dataset while other samples resemble another. Thus, stating the characteristics of images that lead to such bias is beneficial. One apparent bias stems from the size of targets. The evaluation metric of COCO challenges [19], which emphasizes the effects of iconic and non-iconic objects, accounts for the size factor.

Bias in data at the macro level naturally raises a question: does each data sample share equal value or contribution? Recent studies further examine the issue at the micro level, both empirically and theoretically. Focal loss [20] down-weights the loss from well-classified easy samples and focuses more on a sparse set of hard samples. Meanwhile, training with a subset of training samples seems sufficient for training [21]. Koh *et al.* traced the output prediction from a model back to its training data and identified the responsibilities of training samples via influence functions built with gradients and Hessian-vector products [22]. In a follow-up work [23], the weights of training samples were reassigned based on their gradient direction computed from an unbiased validation set. Although research in dataset bias has evolved from qualitative comparison to more qualitative analysis with measurements, the implicit philosophy of learning with data seems unchanged. We keep assuming subconsciously that algorithms should always obey the exact concepts (e.g., object classes) from humans but overlook the possibility that they may require multiple classes to fully comprehend a human-level concept. From an algorithm's perspective, instances in one dataset may be "essentially" different from instances in another, even though they correspond to the same concept according to humans.

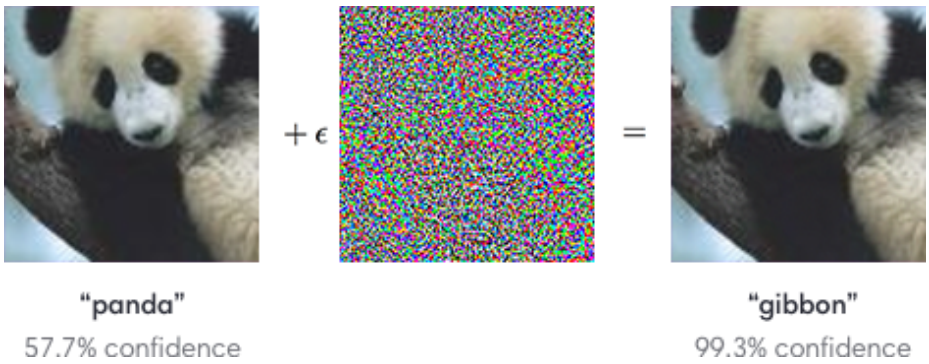


Figure 2.2: An adversarial yet imperceptible perturbation can cause a classifier to misclassify a panda as gibbon [24].

2.3 Adversarial Examples

As an intriguing property first discovered by [9], a hardly-perceptible perturbation on inputs could lead to misclassification by neural networks (Figure 2.2), bringing deep concerns regarding AI safety. Since then, researchers have been working on methods of attacking or defending neural networks.

2.3.1 Adversarial Attacks and Defenses

We can categorize the attacks with multiple orthogonal criteria such as white-box or black-box, targeted or untargeted, one-shot or iterative [25], and image-variant or universal (image-agnostic) [26]. The detailed categories of adversarial attacks are listed at Table 2.1. Among all adversarial attacks, the white-box untargeted image-variant attack is the most fundamental and commonly studied one in literature. More advanced attacks usually contain such a fundamental attack or its variant as a sub-procedure. Therefore, we focus on such white-box attacks to study the rationale of adversarial examples and their implications regarding the image space. Table 2.2 summarizes popular white-box attacks that are often found in benchmarks.

On the defenders' side, two major strategies prevail in current practice [32]: (1) strengthen the classifier such that it accounts for adversarial examples (e.g., adversarial training [24]

Table 2.1: Categorization of adversarial attacks.

Criteria	Category	Details
Knowledge	white-box	Attackers know all details including the weights of the classifier and the defense strategy, and use gradients of the network to generate adversarial examples.
	semi-white	Attackers know all details of the classifier including weights but not the strategy, and use gradients of the network to generate adversarial examples.
	grey-box	Attackers know the architecture of the model and the defence strategy, but not the weights.
	black-box	Attackers knows nothing about the classifier except for the output labels of chosen inputs.
Output labels	untargeted	Adversarial examples are valid as long as they are misclassified.
	targeted	Adversarial examples must be misclassified as the specified category.
Number of steps	one-shot	Attacks take one step in the direction of the gradient
	iterative	Attacks take iterations of steps following the gradient direction
Scope	image-variant	Adversarial perturbations differ from image to image.
	universal	A universal perturbation applies to many images.

[33] and defensive distillation [34]) and (2) detecting and converting adversarial examples back to a normal region (e.g., input transformation [35]). Comprehensive surveys on this topic are available at [36] and Chapter 6 of [37]. Researchers have also established open-sourced benchmarks such as CleverHans [38] and IBM-ART [39] to facilitate studies.

2.3.2 Observations and Explanations

It is worth stating that fooling machine-learning algorithms with adversarial examples is much easier than designing models that cannot be fooled [40]. One can even fool a neural network with very few pixels [41]. In general, research on defending is falling behind the one on attacking. The validity of most existing defending approaches relies on the following implicit assumption: adversarial examples live in different sub-spaces from benign/clean images, and such sub-spaces are detectable (e.g., with metrics such as local in-

Table 2.2: List of popular white-box adversarial attacks

Method	Main ideas, loss functions, or update formulas
FGSM [24], I-FGSM [27]	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
JSMA [28]	Jacobian-based saliency map
DeepFool [29]	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \frac{\lvert f'_{\hat{l}} \rvert}{\lVert \boldsymbol{w}'_{\hat{l}} \rVert_2^2} \boldsymbol{w}'_{\hat{l}}$
Carlini & Wagner [30]	$\min_w \lVert \frac{1}{2}(\tanh(w) + 1) - x \rVert_2^2 + c \cdot f(\frac{1}{2}(\tanh(w) + 1))$
Projected gradient decent [31]	$\boldsymbol{x}_{i+1} = \Pi_{\boldsymbol{x}+\Delta\boldsymbol{x}} \left(\boldsymbol{x}_i + \alpha \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)) \right)$

trinsic dimensionality [42]). Moreover, it's often observed that adversarial examples appear to be much noisier in both image domain and feature spaces. Based on that observation, feature denoising [43] achieved state-of-the-art performance in both white-box and black-box attack settings. However, it is unclear whether the assumption and observation still hold for future attacks, especially as we have insufficient knowledge of the high-dimensional image domain. For efficiency purposes, most existing attacks utilize gradient information of the network and "optimally" perturb the image to maximize the loss function. The noisy pattern might be a side-effect of such a gradient-based operation.

It is still unclear why adversarial examples exist for machine-learning algorithms. In fact, studies on revealing, explaining, and validating the causes of adversarial examples are far less than those that focus on attacks and defenses. Although the research community would agree that studying the causes of adversarial examples is inherently essential, one decisive obstacle is that justifying a relevant hypothesis or statement may often require a thorough examination of the entire proximity of an input sample in a high-dimensional image space. As an efficient "telescope" is not yet available for fully observing the geometry of the high-dimensional image universe, the studies are constraint by the limitation of computation resources. Fortunately, researchers have proposed thoughtful strategies and designs to conduct empirical studies [44, 45], which reveal the characteristics of the learning process and potential reasons for the existence of adversarial examples. In the following

paragraph, we review popular explanations[1] on the causes of adversarial examples:

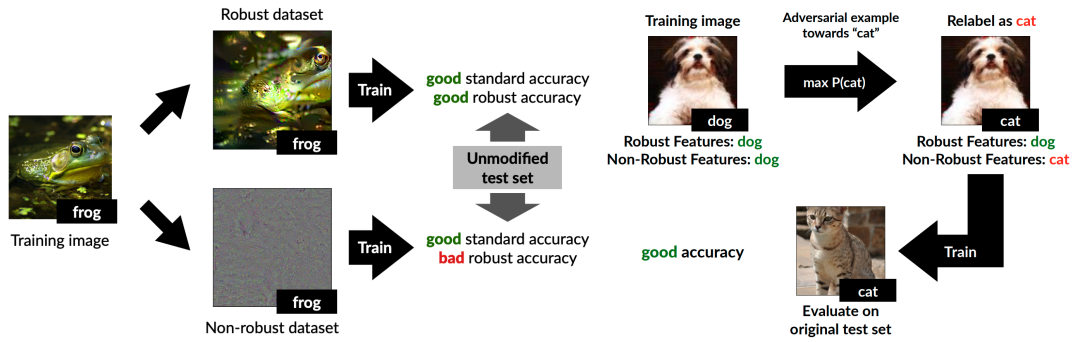
- *Low-probability "pockets" in the manifold*: At the time of first discovery, the authors of [9] interpret adversarial examples as "blind spots" which belong to low-probability "pockets" in the data manifold. To illustrate the situation further, the authors analogize the input domain to the set of rational numbers, which is not dense from a topological perspective.

- *Model linearity*: The current consensus among the research community is that adversarial examples are a product of high-dimensional inputs and the high linearity of the models [24]. The simple and clean explanation states that $w^T(x + \Delta x)$ can differ significantly from $w^T x$, especially when the dimension of inputs x is quite large for natural images.

- *Test error in additive noise*: After observing the relatively high error rates in randomly corrupted image distributions, the authors of [46] argue that it should not be surprising to find adversarial examples. They also suggest that improving adversarial robustness should be aligned with improving robustness against more general and realistic image corruptions.

- *Other geometric explanations*: The boundary tilting perspective [47] states that adversarial examples exist when a decision boundary lies close yet not perfectly aligned to the sub-manifold of sampled data. Thus, the perturbed images are likely to cross the boundary. In [48], adversarial examples are considered a natural consequence of the geometry of $\mathbb{R}^n$ with the L_0 metric. Under the assumption that no category is reserved for "don't know" and the data distribution is not excessively concentrated, it is further proved that adversarial examples are hard to avoid [49].

[1] Due to the extreme difficulty that often occurs for fully justifying (or even designing experiments for) a hypothesis, researchers may raise their views and understandings outside of the major contributions (e.g., in the discussion section of their paper) to avoid potential criticism. Consequently, some of the boldest yet most potential guesses might become buried in literature and thus not included in our reviews.



(a) a frog classifier trained on adversarial noise (b) a cat classifier trained on adversarial dogs

Figure 2.3: Training a decent classifier with adversarial training samples [50].

Over the past a few years, the research community has been regarding adversarial examples as "bugs" or deficiencies of modern AI algorithms. Recently, however, Ilyas *et al.* claim that adversarial examples carry non-robust features that are even generalizable to benign examples [50]. When trained with adversarial examples of dog images with "cat" as target labels, the classifier achieve decent accuracy on benign images of cats (Figure. 2.3b). One can even train a classifier with adversarial examples derived from random noise (2.3a). Thus, the distinctions between benign and adversarial examples start becoming ambiguous. To explain the phenomenon, the authors further divide features into robust and non-robust ones and argue that non-robust features are embedded in adversarial examples. In this work, we further break the binary division of benign and adversarial examples with the proposed detector's dilemma and the technique named "belief distillation."

2.4 Knowledge in Neural Networks

It is commonly assumed that knowledge regarding a particular task can be embedded in the weights of a neural network after training. Knowledge and weights, however, are not in one-to-one correspondence because the same knowledge can be represented by various weight combinations. To leverage or preserve the knowledge learned from previous training, researchers have developed studies in at least three domains: *knowledge distillation* [34], *transfer learning* [51], and *continual learning* [52].

2.4.1 Knowledge Distillation

Inspired by the success of compressing large complex ensembles into smaller faster ones [53], Hinton *et al.* developed the approach by distilling the knowledge in an ensemble of neural networks into a single model for the **same** classification task. In a classic teacher-student framework, the small student network learns to mimic the behaviors of its teacher. From the perspective of the student network, the teacher model's behavior during the inference process contain useful supplementary information to the ground-truth labels in the classification task. In this work, we are interested in the different forms of knowledge representation. According to a summary,[2] one can roughly categorize the forms of knowledge as follows.

Knowledge from logits

Logits and probabilities from the top layer of the teacher network are the most commonly used components for knowledge representation. For example, the supplementary information is defined by the original paper as soft labels computed from the temperature-scaled logits [34]:

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp\left(z_j/T\right)} \tag{2.1}$$

in which z_i is a component in the logits and T is the temperature value.

Knowledge from intermediate layers

Besides outputs from the top layer of the teacher network, features from intermediate layers can also carry knowledge from the teacher. In FitNets [54], intermediate-level hints from the hidden layers of the teacher model are introduced to guide the student model's training process. Single-channel attention maps [55] constructed from the intermediate features can also facilitate the distillation process and result in significant improvement in the student

[2] `https://github.com/FLHonker/Awesome-Knowledge-Distillation`

model's performance.

Knowledge from correlation and mutual information

Apart from instance-level knowledge, the correlation among the inference processes also indicates what the teacher has learned. When training the student model, correlation congruence [56] manages to preserve the sample correlations obtained in the teacher model. In addition, similarity-preserving [57] is proposed to encourage that similar/dissimilar inputs should elicit similar/dissimilar activation patterns in the student's representation space, which is not required to resemble the one from its teacher. In [58], knowledge transfer is formulated as maximizing the mutual information between the teacher and the student networks.

Knowledge distillation has become an active research topic over the past few years since [34], and various representations and combinations of knowledge have been proposed. For a complete list of relevant studies, we refer to active repositories[2,3] on GitHub. By assuming that knowledge is embedded in various representations from the neural networks, existing approaches achieve knowledge distillation without explicitly answering the fundamental question on what knowledge is. In this work, we believe it necessary to address this fundamental yet ultimate question and provide definitions on essential concepts for single-label classification problems, including information, knowledge, beliefs, and truth. By answering the "what-is" question, we evacuate information that used to be overlooked and demonstrate potential benefits from utilizing such information.

2.4.2 Transfer Learning

The goal of transfer learning [51] is to leverage the knowledge from a pre-trained model and apply it for a **different** domain/task. Studies in this field date back to the 90s [59], long before deep learning becomes popular. Transfer learning aims to avoid expensive data-

[3] `https://github.com/dkozlov/awesome-knowledge-distillation`

labeling efforts for training models in a particular domain/distribution by taking advantage of the annotated data and pre-trained models in another domain/distribution. Statistically, the distance between probability distributions can be measured via the maximum mean discrepancy (MMD) [60], with a biased empirical estimate as follows:

$$\mathrm{MMD}_b\left[\mathcal{F}, X, Y\right] = \sup_{f \in \mathcal{F}} \left(\frac{1}{m} \sum_{i=1}^{m} f(x_i) - \frac{1}{n} \sum_{i=1}^{n} f(y_i) \right) \tag{2.2}$$

In the above equation, $X = \{x_i, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$ are observations drawn independently and identically distributed from two distributions, and $\mathcal{F}$ is a class of functions. In transfer learning, especially domain adaptation, MMD is often used as a regularizer in the loss function to ensure domain-invariance of the learned representation [61]. Comprehensive surveys on transfer learning and domain adaptation are available at [62] and [63]. In this work, we provide a geometric interpretation of classification in different domains, even with different levels of features (e.g., semantic vs. aesthetic).

We focus on the application of transfer learning with modern neural networks. In practice, a pre-trained model can either serve as initialization for fine-tuning or a fixed feature extractor [64, 65]. With a fixed feature extractor that is pre-trained on ImageNet [66], a simple classifier with logistic regression (i.e., single fully-connected linear layer) can achieve decent accuracy on various challenging fine-grained classification tasks [67]. Although models trained from random initialization do not necessarily perform worse than their counterparts that are pre-trained on ImageNet [68], they do require longer training time for convergence. For better efficiency and lower costs, "pre-training and fine-tuning" remains a canonical paradigm in computer vision.

2.4.3 Continual Learning

Continual learning studies the ability of a model to learn continually from a stream of data divided by a **sequence** of tasks. Unlike offline training, models in a continual-learning

environment will have limited or no access to training data in previous tasks. With increasing difficulty, continual-learning problems can be roughly categorized into three scenarios: *incremental task learning*, *incremental domain learning*, and *incremental class learning* [69, 70]. In an incremental-task-learning scenario, multiple classifiers (i.e., heads), each corresponding to a task, are trained on top of a shared feature extractor. At test time, the task is specified, and prediction will be made by the classifier associated with the task. In contrast, the classifier's width in incremental class learning will keep growing as new categories arrive. At test time, no clues will be provided regarding the task to which an input belongs. Among these three scenarios, the incremental class learning aims at matching the highest standard from offline training. Moreover, it is more realistic because the tasks will not be specified in practice.

One of the biggest issues in continual learning is catastrophic forgetting, meaning that a model will forget responses that are learned previously once the input stimuli disappear. One popular explanation on catastrophic forgetting assumes that knowledge is embedded in a model's representational resources (e.g., weights of neurons in a neural network). Therefore, catastrophic forgetting occurs when significantly different data in subsequent tasks force the model to overwrite previously learned knowledge through updating the shared representational resources [10, 71]. On the contrary, the knowledge can be well-preserved in an off-line training scenario by revisiting the samples multiple times throughout the training process. Following the explanation, researchers have developed various approaches to mitigate catastrophic forgetting, among which memory replay and update regularization are two canonical strategies [72]. Memory replay keeps a small batch of samples from previous tasks in the storage buffer, and those samples are available in subsequent tasks. In update-regularization methods, the weights of the classifiers are updated in a constrained fashion to preserve the consistency of predictions for previously seen samples. Technically, knowledge distillation [34] is often applied to preserve knowledge learned from previous tasks. In such a teacher-student framework, models at later tasks act as students whose goal

is to match the teacher predictions from themselves at early tasks. For incremental class learning scenarios, methods with memory replay of samples from previous tasks outperform those without memory replay by a large margin [70]. A comprehensive survey on continual learning and catastrophic forgetting is available at [72].

2.5 Image Style Transfer

2.5.1 Artistic Neural Style Transfer

Previous studies on image style transfer are dominated by destructive methods that directly manipulate each pixel. Gatys *et al.* first demonstrated that content and style of images are separable by measuring style with Gram matrices of intermediate features extracted a neural network [7]. The pioneering work on neural style transfer has opened a gate for AI in artistic applications. Subsequent studies managed to enhance computation efficiency [73, 74] and extend the range of style domains [75, 76]. With the help of GANs [15] Researchers have formulated style transfer as image-to-image translation between two domains, both in paired [77] and unpaired [78] modes. With the state-of-the-art StarGAN [79], one can simultaneously translate inputs into multiple domains.

Destructive methods work ideally for transferring between well-distinguishable domains that differ significantly in strokes and texture (e.g., paintings). As for photographs, however, they are hindered by severe drawbacks. Although fantastic artwork has been produced, the neural-network methods remain as black boxes that cast little light on the process. Sometimes, the failure cases are neither predictable nor explainable. Unlike paintings and images that belong to various domains, photographs are considered samples within the same domain, leading to very little tolerance on distortion for high-quality production.

2.5.2 Photo-Realistic Style Transfer

Despite impressive results from artistic style transfer, much less success has been reported in the photo-realistic domain until most recently [80, 81, 82]. To fill in the blanks for

non-destructive photo editing, Hu *et al.* proposed a groundbreaking white-box framework *exposure* [80], which automates the photo retouching process using human-understandable filters and curve operations. The framework incorporates GANs with deep reinforcement learning (RL), a state-of-the-art class of techniques that outperforms humans on various strategic games [83]. Unfortunately, both GAN and RL are hard to train in practice. In [81] [82], styles are transferred via closed-form whitening and coloring transformations (WCT) of pixel distributions, and an exemplar photo is required for specifying the target style. In addition to image translation, relevant studies are often observed in the field of photo enhancement (e.g., [84]), the goal of which is to correct a photo from a problematic stage. Photo-enhancement methods mostly bring a photo from problematic to normal but provide insufficient support for navigating among suitable styles that are already normal.

Existing methods on style transfer bring limited insights and comprehension on styles with respect to photography. Compared with artistic and texture-based styles, photo styles are far more abstract to comprehend and less straightforward to quantify. One major difference lies in the underlying assumption that the content is somehow predefined and normally irreplaceable when considering photo styles. We believe that the first step towards automating expert-level non-destructive style transfer is to comprehend photo styles in general. In this work, we revisit the photo style recognition problem, which was established before the deep-learning era.

2.5.3 Photo Style Recognition

The first large-scale image-style recognition benchmark is established from the AVA dataset [85], in which a multi-label classification is formulated to recognize 14 image styles. In follow-up studies, however, the AVA dataset is mainly used for aesthetic rating and image retrieval. Single-label photo-style recognition has also been benchmarked on the Flickr-Style-80K dataset [86], containing 80K images covering 20 photo styles. Unlike content-based image recognition benchmarks (e.g., ImageNet [66]), image style recognition is far

from solved because the follow-up work is considerably less, and the classification accuracy is much lower. One possible explanation is that compared to semantics, photo styles are more advanced and abstract concepts. Another possible explanation is that existing benchmarks tend to mix various levels of styles, including composition, subjects, and color distributions, which might confuse the classifier during training. In addition, existing benchmarks are not built in a controlled manner as their samples differ from one another both in content and style. As these two elements are usually correlated, the classifier can often predict styles according to contents, which leads to concerns on the validity of the learned representation.

Besides the above large-scale datasets for photo style recognition, the MIT-Adobe FiveK dataset [87] contains five thousand RAW photos retouched by five photographers. In other words, each content image has five styles named after the retoucher. The remaining difficulty, however, is that those styles from retouchers are too flexible for a classifier to capture. Thus, to further disentangle the color-distribution photo style from other image styles, we apply various non-destructive artistic presets to the retouched (i.e., not problematic) photos in the FiveK dataset, generating photos with various *consistent* styles yet same contents. Those photos are later used in benchmarking photo style recognition.

One additional factor that may affect the photo-style recognition is the usage of color spaces. As an interesting topic in the cross-domain of art and science, color space addresses the concern on how to organize color elements (i.e., tuples) scientifically and effectively. Different organizations then lead to different distances between samples in the image space, which could potentially result in different outcomes of the same algorithm. Motivated by the observation that content/semantic classifiers reach different accuracies when trained on different color spaces [88] [89], we also investigate the impact of color spaces in the context of photo styles.

2.5.4 Generative Adversarial Networks

Generative adversarial networks (GANs) serve as the core engine of image-translation approaches for style transfer. Introduced by Goodfellow *et al.*, generative adversarial networks [15] are unsupervised learning algorithms consisting of two modules, namely a generator G and a discriminator network D. By contesting with each other in a two-player minmax game, the generator is expected to produce images that appear to be authentic and thus cheat the discriminator, whose role is to judge whether the output comes from real data or not. Mathematically, the value function V of a original GAN can be written as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.3}$$

where $G : \mathbb{R}^m \to \mathbb{R}^n$ maps a noise sample to an authentic image, and $D : \mathbb{R}^n \to \mathbb{R}$ outputs the probability that an image comes from true data rather than G. The variables $\boldsymbol{x}$ and z denote a data sample and a input noise variable, respectively. Under these settings, the generator G learns a mapping from an input in the noise distribution $p_z(z)$ to a generated distribution $p_g(\boldsymbol{x})$ which equals the true data distribution $p_{\text{data}}(\boldsymbol{x})$ at the global minimum of the value function V. The optimal generator G^* minimizes the divergence between p_{data} and p_g. Besides the original value function which is associated with the Jensen-Shannon divergence, Nowozin *et al.* derived additional value functions and generalized them with arbitrary f-divergences [90].

Although GANs have achieved great success generating photo-realistic images, they are known to be unstable and hard to train. Researchers have been consistently designing enhanced versions of GANs, such as WGAN [91], LSGAN [92] and BEGAN [93]. None of these, however, consistently outperforms the original one according to large-scale comparisons [94]. The *exposure* framework for global photo adjustment adopted one of the most popular variants named WGAN-GP [95], which measures the distance between p_g and p_{data} with Wasserstein distance (also called earth mover's distance) and append a

penalty on the gradient norm if it moves away from its target value 1. The value function L is then written as:

$$L(D, G) = \mathbb{E}_{\tilde{\boldsymbol{x}} \sim p_g}[D(\tilde{\boldsymbol{x}})] - \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[D(\boldsymbol{x})] + \lambda \, \mathbb{E}_{\hat{\boldsymbol{x}} \sim p_{\hat{\boldsymbol{x}}}}[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2] \tag{2.4}$$

where $p_{\hat{\boldsymbol{x}}}$ is sampled uniformly along straight lines between pairs of points sampled from the data distribution p_{data} and the generated distribution p_g.

In the subsequent chapters, we will revisit GANs in different contexts. Directly applications of the technique are concentrated at Chapter 5 for photographic style recognition and stylization. We first study the color space transformation at the image level by formulating it as image translation with GANs. We then enrich the inputs of the original exposure framework with artistic features, which simplifies the judgment process of the discriminator D. The discussion regarding the rationale behind GANs is raised in the ability of the network (3.3.3). Instead of dividing them into generators and discriminators, we argue that the two abilities co-exist in a discriminative classifier.

CHAPTER 3

CONCEPTUAL MODEL FOR NATURAL-IMAGE SPACES

In this chapter, we manage to explain the robustness issues in neural networks with a proposed conceptual and hypothetical model for natural-image spaces. Under the unified view of machine learning from variational calculus described in Section 3.1, various computer-vision tasks can be interpreted as optimizing a functional against different independent variables under different constraints. Among all determinants in such an optimization process, data themselves are within the most crucial ones whose characteristics are worth studying. In the proposed hypothetical image-space model in Section 3.2, we analyze the properties of the natural-image spaces and their consequences on the performance of algorithms. By formulating the classification task as a partition of the image space, in Section 3.3, we further define fundamental concepts regarding knowledge for AI and machine perception.

3.1 Variational-Calculus View of Machine Learning

We analyze the high-level scheme of machine learning from the view of variational calculus and reveal potential reasons for robustness issues in neural networks. In general, the goal of the supervised learning is to obtain an approximation of the underlying target function $f(\cdot)$ by optimizing an objective loss function $L(\cdot)$ with the idea of gradient descent [96]. According to the geometric view taken in functional analysis, the target function $f(\cdot)$ is an extremum point, whose values may be explained as components of an infinite-dimensional vector indexed by its domain X (i.e., $\{f_x\}_{x\in X}$). Meanwhile, the objective loss function $L(\cdot)$ corresponds to a functional $L(f(\cdot),\cdot)$ in calculus of variations. A functional is a special type of function that takes another function as its input. A simple example of functionals can be

expressed as

$$L(f(x), x) = \int_a^b (f(x) - y(x))^2 \, dx,$$

in which $y(x)$ is given. We can discretize the expression and obtain an objective loss function (i.e., mean-squared error) that are often used when training neural networks:

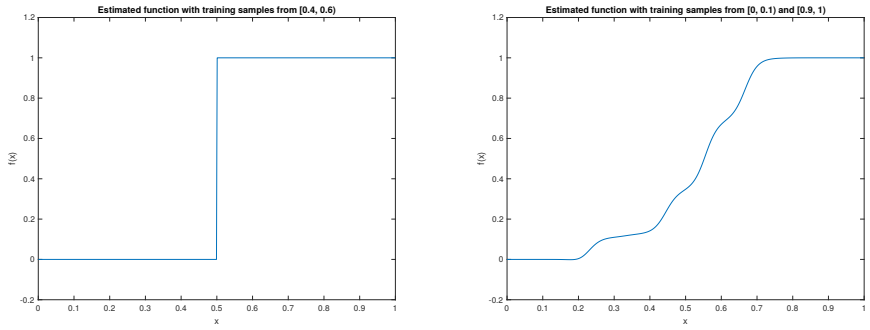
$$\text{MSE}(f(x), x) = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2 .$$

Under the unified scheme from variational calculus, we can easily categorize relevant computer-vision tasks as follows:

- Supervised learning: During the training process, data $x \in X$ is fixed, and the function f is updated to minimize the loss L.
- Adversarial attacks and neural style transfer: After training, the approximated function f is fixed, and each piece of data x is updated to meet the requirement specified by a new loss L'.

The variational-calculus view explains that the training process in supervised machine learning is heavily data-dependent because the output of the objective loss function (i.e., a functional) depends on not only the *summand* $f(\cdot)$, but also the *interval* (i.e., the data x) for summation. Therefore, deficiencies in data themselves and data usage will impact learning-based algorithms. Such deficiencies on data x, however, cannot be mitigated by improving optimizers or network structures for $f(\cdot)$.

We then introduce the *point-function duality* of the underlying target function f to emphasize that the training process is under-constrained. As an optimal point of the loss functional $L(f, x)$, the target function f cannot determine its domain and range. As a function, the topological properties of its actual domain and range matter, especially if we apply $f : X \to Y$ to other test inputs. Different from admissible functions in variational calculus, the approximated target function sets no restrictions on the value of its inputs and out-



(a) Training with w/ 40,000 samples from $[0.4, 0.6)$

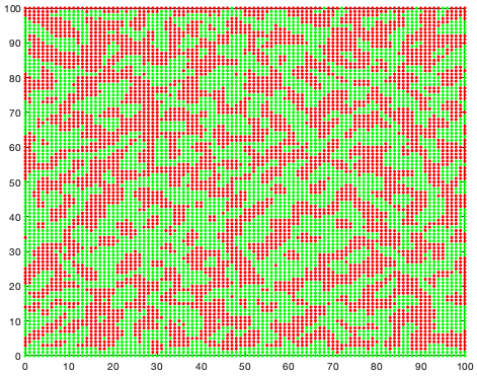
(b) Training with 20,000 samples from $[0, 0.1)$ and 20,000 from $[0.9, 1)$

Figure 3.1: Training samples affect the geometry of the approximated space, demonstrated using linear regression on $f(x) = \text{sign}(x - 0.5)$ with different training data.

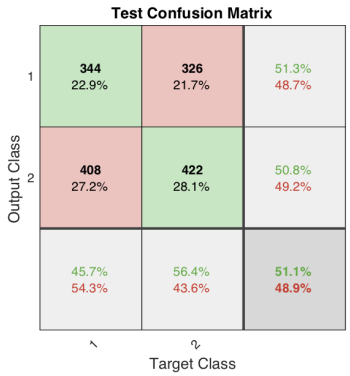
puts, and might be locally valid around training data. If the function is trained using data within a certain neighborhood, dataset bias can occur at other regions. Moreover, as Figure 3.1 shows, the shape of learned curves and boundaries is determined by training data and sometimes cannot be controlled directly by learning algorithms. Similarly, the decision boundaries of image classifiers will shift as training samples differ. Therefore, the dataset bias is an intrinsic property stemmed from the point-function duality of target functions.

3.2 Hypothetical Image-space Model

As the learning process is heavily data-dependent according to the variational-calculus view of machine learning, we deem that the characteristics of the input space are worth studying. Another motivation is that unlike decision trees, neural networks cannot classify even and odd numbers (Figure. 3.2), which implies that the topological properties of the input space matter. In this section, we study the topological and geometric property of the natural-image spaces, as well as their consequences on computer-vision algorithms using neural networks. To match the discrete settings in image storage, we address natural-image spaces in a completely discrete manner. Let $\mathbb{Z}_{[l,u]}$ denote the set of integers from l to u. Given an image I with resolution $w \times h$ and d channels, the image is considered as a point of the $\mathbb{Z}_{[0,255]}^{w \times h \times d}$- based natural-image space $\mathbb{I}^{w \times h \times d}$ if I appears to be natural according to a perfect



(a) Visualization of the classification



(b) Confusion matrix on test set

Figure 3.2: Classifying the sum of two integers into even and odd with neural networks

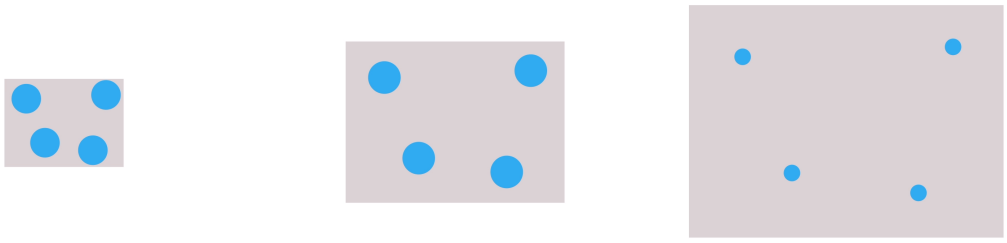
discriminator. We assume that a perfect discriminator is accessible (e.g., using human judgment). The essence of the concept is twofold: (1) there exists an image space for each resolution, and (2) those image spaces are not necessarily dense at all times.

3.2.1 Properties of Natural-image Spaces

In this subsection, we will describe what discrete natural-image spaces look like by listing important properties. We further demonstrate that these properties can be closely related to data usage in machine learning.

Sparsity and the Scale-space Effect

A natural-image space is obviously sparse as the probability that a random sample in $\mathbb{Z}_{[0,255]}^{w\times h\times d}$ belongs to $\mathbb{I}^{w\times h\times d}$ is small. When resolution increases, the total number of cases increases exponentially. Similar to the effect in scale-space theory, flaws in higher resolutions tend to be identified more easily by humans. Hence, natural-image spaces are denser in lower resolutions because the quantity of valid natural images increases more slowly than that of possible cases (Figure 3.3). Appendix A provides a detailed reasoning process that supports the argument. The scale-space effect is consistent with results that show algorithms for image generation and translation produce more natural in images with relatively lower resolutions. In image classification, however, such properties are not appropriately ad-



(a) Low resolution (b) Med resolution (c) High resolution

Figure 3.3: Illustration on the scale-space effect of image spaces

dressed, especially with a limited number of categories provided. Under current settings, a classifier assumes that the underlying target function is well-defined everywhere and learns to color the entire input space with a fixed number of colors. When the classifier eliminates all impossible categories, it has to pick the remaining one as the final output. Unfortunately, no category for exceptions exists. To classify the input space continuously and smoothly, the decision boundaries are often deformed. As the first step for handling exceptions, we propose to augment training samples with an extra random-noise category with the hope that it would fill in the invalid regions among categories and push the decision boundaries towards the centroid of each cluster.

Connectivity

Based on the sparsity property, we infer that two images belonging to the same category are not always "path-connected" through adjacent grid points in $\mathbb{Z}_{[0,255]}^{w\times h\times d}$. At the micro level, a natural-image space can be regarded as a *quotient space* $Y = X/\sim$ consisting of numerous equivalence classes of "path-connected" images that can be derived from each other (e.g., using filters). These equivalence classes, however, are not necessarily connected. One gap exists between a labeled natural-image space (i.e., a quotient space) and the one approximated by a neural network because the regions classified by the network tend to be connected [44]. In addition, the gap is exacerbated by a fixed yet insufficient number of categories. The setting reflects a subconscious assumption that has been kept since

the beginning of machine learning: Algorithms should always learn and follow the exact concepts (e.g., object classes) specified in a particular dataset. In fact, they may require multiple classes to fully comprehend a human-level concept. It is possible, however, that instances in one dataset are "essentially" different from instances in another, even though they correspond to the same concept according to humans.

3.2.2 Reinterpretation of Existing Work

At the end of this section, we explain existing work from the viewpoint of the conceptual image-space model. The essence of the explanation is to refresh our understanding towards experiment results from a unified yet nonexclusive perspective. The following opinions are provided as alternative and supplementary rather than a correction of the original claims. While image translation can be interpreted as finding a mapping between two points in a natural-image space, image generation seeks the reverse mapping from a latent space to a natural-image space. The major difficulty lies in need for designing an appropriate loss functional that indirectly regulates the performance of its extremum (i.e., the target function). Effective approaches are found in recent studies such as those in [77] and [78]. Specifically, L_1 norm forces the image to be sufficiently sharp, which prevents the output from leaving a natural-image space. Strict rules for coloring adopted by the facade dataset [97] limits the cases of outputs, leading to a denser output space. Cycle loss manages to ensure the bidirectional property of the mapping so that it obtains a more stable bijection. In the progressive growing of GANs [5], the network first locates natural-image clusters at lower resolutions, which is comparatively easier as the space is denser. With these coarse locations, outputs in higher resolutions are more likely to stay in natural-image spaces. The network then grows the scale progressively to refine outputs for better details. In [98], stability-loss guarantees that the target function is locally consistent within the neighborhood of each image, providing buffer zones between images and decision boundaries. The authors of [99] proposed a subnetwork that distinguishes benign data from perturbed ones,

shrinking the natural-image spaces for the subsequent classifier. By assigning weights to training data, RetinaNet [100] detects objects more accurately at a fast speed.

3.3 Topological View of Knowledge for Single-Label Classification

In this section, we define knowledge from the topological perspective and elaborate potential benefits that result from such a viewpoint. In classic epistemology, knowledge is defined as a justified true belief[1]. Formally, an agent S knows that a preposition P is true if and only if:

1. P is true,
2. S believes that P is true, and
3. S is justified in believing that P is true.

The sufficiency of the definition, however, is challenged by counter-examples raised by the Gettier problem [101] in that the reasons for the belief, even though justified, might still be false. Fortunately, the concern is not a restriction for understanding machine knowledge under the assumption that the ground truth is guaranteed, and inference processes (i.e., forward passes) for samples are independent of one another. We now define knowledge for AI in the context of image-space partition.

3.3.1 Definitions

Let us reconsider the single-label classification as a space-coloring problem in which a classifier paints every data sample in the input space with a unique color (i.e., the label). Without loss of generality, a combination of labels in multi-label classification can be renamed with a new label. For simplicity, we assume that all inputs in an image space share a common shape, which is consistent with configurations in most, if not all, neural-network classifiers. Thus, an image space X with dimension d (e.g., the product of height, width,

[1] seemingly by Plato, according to [101]

and depth) is a subset of $\mathbb{R}^d$ (real tuples) or more precisely $\mathbb{Z}^d_{\geq 0}$ (non-negative integer tuples). A classifier $f : X \ni x \mapsto y \in Y \subset \mathbb{Z}_{\geq 0}$ then colors the entire image space by assigning a scalar non-negative integer label y for each input image x.

Information

We define a piece of information (directly related to a classification task) as a pair of image and its label (x, y). Information from a classifier f is then *any* pair of input image and output label (x, y) that satisfies $y = f(x)$. It is worth noting that x is not necessarily a meaningful natural image. The image can be derived from any samples in the input space, such as prototypes, centroids, mixtures, or even noise. In addition, the label can be either correct or wrong, as a piece of information may either be useful or misleading.

Knowledge

We explicitly define knowledge as a collection of accessible pairs of the input image and output label $\{(x, y)\}$ in which the output label matches the justified ground truth provided in training data of a particular task. The definition is originated from the concept of "justified true belief" in epistemology. One may argue that knowledge should be associated with a subject entity (e.g., a network) according to previous studies such as knowledge distillation [34]. In this work, we further clarify the so-called "knowledge from a network" as beliefs, which will be defined below.

Belief

Different from knowledge, a belief should always be associated with a subject entity (i.e., a classifier). We define a belief of a classifier f as an input-label pair $(x, f(x))$ that follows the prediction of the classifier f. All beliefs of the classifier f then constitute a partition $\{C_k\}_{k \in \{j \in \mathbb{Z}:\ 0 \leq j < N\}}$ of the input space X based on the predicted labels from the classifier. Each subset C_k in the partition is an equivalence class consisting of samples that satisfy $f(x_i) =$

$f(x_j)$ for $\forall x_i, x_j \in C_k$. Following the quotient-space model in Section 3.2.1, the partition can be explained as a collection image-label pairs $(x, f(x))$ grouped by the equivalence relation f. By definition of the partition, the following conditions should hold for $\{C_k\}$:

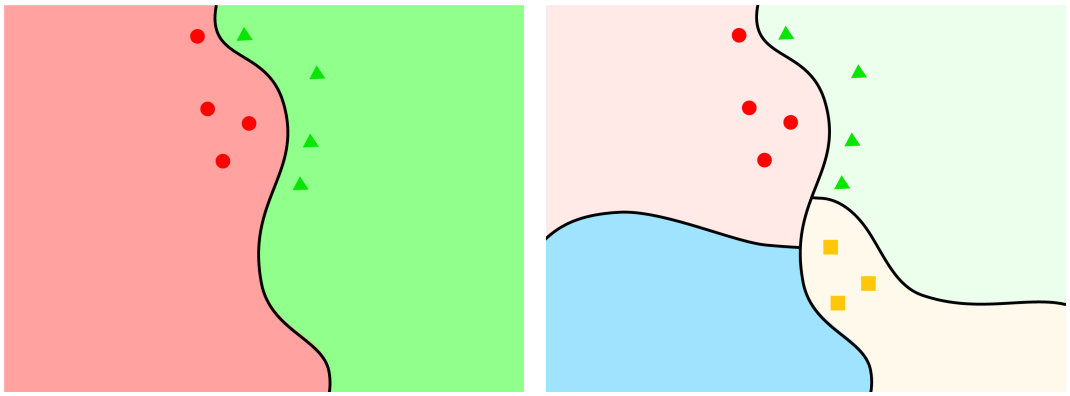
- $\emptyset \notin \{C_k\}$: the subsets don't contain empty sets.
- $\bigcup_k C_k = X$: the union of the subsets should cover X.
- $C_i \cap C_j = \emptyset$, for $\forall i \neq j$ and $0 \leq i, j < N$: the elements in $\{C_k\}$ are pair-wise disjoint.

Truth

Similar to beliefs, the truths of single-label classification is defined as input-label pairs $\{(x, g(x))\}$ following the outputs from the ground-truth (or oracle) classifier g. In other words, the truths are special beliefs from the oracle. Correspondingly, the truths form a partition $\{T_k\}_{k \in \{j \in \mathbb{Z}:\ 0 \leq j < K\}}$ of the input image space X. As a special case, the above three conditions still hold for the partition. However, there is a gap between the partition derived from the beliefs of a common classifier and that from the oracle. We will address the gap in the upcoming subsection.

3.3.2 Beliefs vs Truths

As illustrated by Figure 3.4, two crucial differences exist between partitions from beliefs of a classifier f and the truths according to the oracle g. One difference is that the numbers of subsets differ (i.e., $K \gg N$) in most if not all cases. Consequently, when a classifier f has eliminated all categories which are impossible, then whatever remains must be the prediction. The other is that C_k tends to be path-connected [44], whereas T_k may be equipped with an arbitrary topology. We believe the second one results from the fact that neural networks can only approximate continuous functions according to the Universal Approximation Theorem [102]. Moreover, the geometry of the partition from beliefs can be exacerbated by a fixed yet insufficient number of categories. To connect separated samples, a network clas-



(a) Beliefs of an ordinary classifier f: 2 path-connected subsets (circles and triangles belong to the surrounding category)

(b) Truths from the ground-truth classifier g: 7 subsets (circles and triangles differ from the surrounding category)

Figure 3.4: Illustration on the differences between partitions from beliefs and truths: both classifiers "perfectly" assign labels for red circles and green triangles, leading to zero error. The partitions of the input space, however, may differ significantly.

sifier may need to deform its decision boundaries, creating somewhat arbitrary bridges that pass through uncertain regions in between.

Truth be told, a classifier f will be considered "perfect" (i.e., zero-error) as long as $T_k \subset C_k$, for $\forall k < N$ is satisfied, assuming the indices of categories are matched. The subset relation indicates that all instances in a category shall be gathered. Meanwhile, as the subsets $\{C_k\}$ are pair-wise disjoint, no confusion shall exist among learned categories. Therefore, both classifiers in Figure 3.4 are considered "perfect" with respect to red circles and green triangles in terms of classification accuracy. The abilities of the two classifiers, however, may differ significantly. In the following subsection, we describe two abilities of trained classifiers.

3.3.3 Two Levels of Ability

After training, a classifier is equipped with two levels of abilities: *differentiation* and *imagination*. These two abilities are embedded into the weights of the neural-network classifier. Weights and abilities, however, are not completely equivalent because various weight combinations may achieve similar abilities.

Differentiation

Differentiation refers to the ability to predict the output label for a given input image. It is a deterministic process that directly applies the approximated classifier f to any input x. Such a process is considered an imitation of the decision-making or judgment process in human brains.

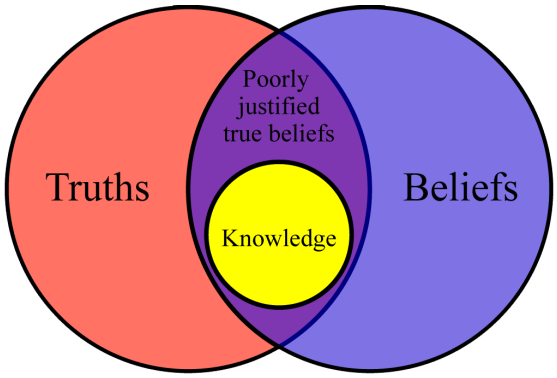
Imagination

Imagination is the ability to produce unseen pairs of the input image and output labels based on the classifier's beliefs. It is a generative process that starts from an arbitrary point in the input space and navigates to the destination with high confidence to belong to a category. The process can be viewed as an imitation of imagining or sketching an instance for a category. In practice, adversarial examples happen to demonstrate the imagination ability of a classifier. Different from previous studies which tend to treat the two abilities separately, we argue that the imagination ability can be valuable even for discriminative networks because imagination provides hidden information from the classifier. We will elaborate on the hidden information in the following section.

3.3.4 The Hidden Information

When considering classification as a partition of the input space, the primary (sometimes the only) goal is to minimize the error from training data by grouping them into correct categories. During training, the classifier learns knowledge from training data and their ground-truth labels. It is worth pointing out that the entire input space has been simultaneously partitioned, even though ground-truth is unavailable at most places. The partition reflects the beliefs of a classifier, regardless of the correctness of the predicted labels. Among all types of information illustrated at Figure 3.5, two of them are often underutilized in previous studies: *poorly justified true beliefs* and *false beliefs*.

The *poorly justified true beliefs* refer to the correct, accessible input-label pairs that



Truths: $\{(x, g(x)) \,|x \in X\}$

Beliefs: $\{(x, f(x)) \,|x \in X\}$

False beliefs:
$\{(x, f(x)) \,|x \in X, f(x) \neq g(x)\}$

Knowledge:
$\{(x, f(x)) \,|x \in X_{\text{acc}} \cap X_{\text{task}}, f(x) = g(x)\}$

Poorly justified true beliefs:
$\{(x, f(x)) \,|x \in X_{\text{acc}} \setminus X_{\text{task}}, f(x) = g(x)\}$

Figure 3.5: Euler diagram representing the definition of knowledge [103].

are beyond the scope of the current task, that is, $\{(x, f(x)) \,|x \in X_{\text{acc}} \setminus X_{\text{task}}, f(x) = g(x)\}$. Those inputs, moreover, are not required to be instances of categories within the training set. When training classifiers with a limited number of categories in existing benchmarks, the classifiers may take advantage of all possible and efficient "descriptors" that facilitate distinguish those categories. The "descriptors" are equivalent to features in human brains if and only if the categories in the classifier's belief are identical to those in the truths. To bridge the gap, we propose a methodology in Section 4.2 that compensates for the missing true beliefs by training with more categories than necessary.

The *false beliefs* are input-label pairs in which the labels do not match the ground truth, that is, $\{(x, f(x)) \,|x \in X, f(x) \neq g(x)\}$. Despite the incorrect classification, false beliefs can still reflect the partition derived from the entire beliefs. When the number of categories is fixed, we may assume that all beliefs of a particular classifier are correlated because the subsets tend to be path-connected (Figure 3.4a) and the sum of probabilities equals 100%. Therefore, an alternative yet non-exclusive explanation on training a decent classifier with adversarial examples [50] is that those samples agree and are also in high confidence with the partition according to the base classifier (i.e., the one used for crafting adversarial examples). In that case, part of the true beliefs can be distilled from the base classifier even though the labels for adversarial training examples are inexplicable to humans. In Section 4.3, we further demonstrate that benign and adversarial examples are not irreconcilable

and analyze how adversarial examples can help boost classification accuracy on benign examples.

3.4 Summary

As a complement explanation to previous work which primarily formulates tasks from a pure statistic perspective, in this chapter, we provide a geometric and topological view of machine learning (supervised in particular) with the proposed hypothetical image-space model. Instead of injecting human understanding and knowledge into neural networks, we put ourselves **in their place** and define fundamental concepts associated with their perception process. The essential takeaways from this chapter include:

- Machine-learning tasks can be interpreted under the unified variational-calculus view as optimizing a functional against different independent variables under different constraints. In addition, the target function approximated by a neural network can be regarded as an infinite-dimensional vector indexed by its domain.

- The space of natural images under a particular resolution is a sparse quotient space consisting of numerous equivalence classes. The space becomes even more sparse as resolution increases, and the equivalence classes are not necessarily path-connected.

- By formulating image classification as a partition of the image space, one can define fundamental concepts of machine perception (i.e., information, knowledge, beliefs, and truth) with input-label pairs.

- Two crucial differences exist between the partitions from beliefs of a classifier and truths: the number of categories and path-connectedness of the regions. Those two differences are the root reason for robustness issues in neural networks.

- Because the entire input image space has been partitioned simultaneously, a neural-network classifier has capabilities for both differentiation (discriminative) and imag-

ination (generative). Among all types of information in the image space, two types of hidden information, poorly justified true beliefs and false beliefs, used to be overlooked in previous studies.

At the end of this chapter, we state the limitation of the topological view of knowledge at the current stage. For conciseness and clarity purposes, the classification inference is currently defined as a scalar-value function whose range is specified as the non-negative integers (i.e., the output label). The soft probability (or logits) from a classifier, however, contains more *evidence*[2] related to the data sample than just the class label. Thus, it is reasonable to redefine the inference process at Section 3.3.1 as a mapping $\hat{f} : X \ni x \mapsto y \in Y \subset \mathbb{R}^N_{[0,1]}$ which maps an image to a probability vector. Fortunately, one can easily achieve this by generalizing the equivalence relation $\sim$ of the quotient space in Sections 3.2.1 and 3.3.1 to $\hat{f}$. Two concerns, however, are resulting from such a generalization. One concern is that the falsifiability of a piece of information is lost. On the one hand, truths are hard to define with N-tuple or K-tuple of soft real values in case that the one-hot encoding is not preferred. On the other hand, it becomes almost impossible to match the exact truth vector with the output from the network. The other concern is that number of equivalence classes in the quotient space becomes uncountably infinite, and the cardinality of the set is significantly increased, even though they are later mapped into a finite set of scalar values.

In the following two chapters, we validate the proposed hypothetical image-space model and illustrate the benefits from the philosophy brought by the topological view of knowledge in the two aforementioned concrete applications: single-label classification (Chapter 4) as well as photographic style recognition and photo-realistic transfer (Chapter 5).

[2]Following the definitions in Section 3.3.1, supportive evidence for a piece of information $(x, f(x))$ can be defined as input-output pairs $(x, f_0(x))$ such that $f(x) = f_1 \circ f_0(x)$, which covers various forms of knowledge representations reviewed in Section 2.4.

CHAPTER 4

IMAGE-SPACES IN SINGLE-LABEL CLASSIFICATION

Our understanding of learning in natural-image spaces indicates the uncertain impact of training samples and the potential benefits of utilizing the topological and geometric properties of the input space. In this chapter, we design experiments to validate our hypothetical image-space model in the application of single-label image classification. We start with controlled experiments that demonstrate the impact of training samples within a single dataset in Section 4.1. Then in Sections 4.2 and 4.3, we elaborate the usage of the two hidden information under our topological view of knowledge, that is, poorly justified true beliefs and false beliefs. The usage of adversarial examples in those sections implies that benign and adversarial examples are not irreconcilable. Section 4.4 presents an adaptive of adversarial robustness and brings to light the variance of robustness among categories and samples. Finally, we verify the geometric causes of adversarial examples.

4.1 Impact of Training Samples

In this section, we demonstrate the impact of training samples by retraining shallow networks provided by MATLAB[1] on subgroups of the training data in CIFAR-10 [104]. A regular training process terminates at an early stage before over-fitting occurs. After training, we categorized the training samples into two super-classes: those that were correctly classified and those that were misclassified. Within each super-class, we sorted the samples according to the maximum probability score, denoted by "confidence" for correctly classified samples, or "illusiveness" for the misclassified. From the correctly classified samples, we selected two subgroups with relatively higher (S_{hc}) and lower (S_{lc}) confidences. Similarly, two subgroups with higher (S_{hi}) and lower illusiveness (S_{li}) were selected from the

[1] mathworks.com/help/vision/examples/object-detection-using-deep-learning.html

Table 4.1: 50-time average performance of the classifier retrained with subgroups of CIFAR-10 training data. S_{c}^{p}: the top p percent of training samples selected with criteria c. Evaluation metrics are average test accuracy (A), standard deviation of test accuracy (σ_A), average confidence of correctly classified samples (P_c), average illusiveness of misclassified samples (P_i), average probability of the ground-truth category for misclassified samples (P_g).

Training set	A	σ_A	P_c	P_i	P_g
$S_{\mathrm{lc}}^{.25} \cup S_{\mathrm{li}}^{.25}$	27.12%	2.40%	26.56%	22.91%	14.65%
$S_{\mathrm{lc}}^{.25} \cup S_{\mathrm{hi}}^{.25}$	19.26%	0.92%	24.68%	22.86%	13.67%
$S_{\mathrm{hc}}^{.25} \cup S_{\mathrm{li}}^{.25}$	53.61%	1.23%	79.67%	58.00%	12.73%
$S_{\mathrm{hc}}^{.25} \cup S_{\mathrm{hi}}^{.25}$	51.49%	1.29%	66.57%	44.77%	14.57%
$S_{\mathrm{lc}}^{.50} \cup S_{\mathrm{li}}^{.50}$	52.16%	1.58%	45.73%	36.39%	18.61%
$S_{\mathrm{lc}}^{.50} \cup S_{\mathrm{hi}}^{.50}$	41.07%	1.18%	35.11%	30.86%	17.99%
$S_{\mathrm{hc}}^{.50} \cup S_{\mathrm{li}}^{.50}$	65.78%	0.58%	88.08%	70.23%	11.41%
$S_{\mathrm{hc}}^{.50} \cup S_{\mathrm{hi}}^{.50}$	60.48%	0.67%	75.62%	50.50%	15.92%
$S_{\mathrm{lc}}^{.75} \cup S_{\mathrm{li}}^{.75}$	68.76%	1.16%	74.25%	52.34%	18.36%
$S_{\mathrm{lc}}^{.75} \cup S_{\mathrm{hi}}^{.75}$	60.55%	1.31%	58.33%	43.00%	20.33%
$S_{\mathrm{hc}}^{.75} \cup S_{\mathrm{li}}^{.75}$	70.81%	0.31%	92.67%	76.09%	10.28%
$S_{\mathrm{hc}}^{.75} \cup S_{\mathrm{hi}}^{.75}$	70.85%	0.47%	80.45%	55.76%	16.24%

misclassified. With all subgroups the same size, we then retrain the network with the selected subgroups to illustrate the impact of different training samples. For each experiment, we obtain results by averaging 50 executions. The performance measurements include average (A) and standard deviation (σ_A) of test accuracy, average confidence (P_c) and illusiveness (P_i) of the prediction for correctly and incorrectly classified samples, as well as the average probability assigned for the ground-truth category (P_g). We further assume that the output probability of each category is negatively correlated with the distance from the sample to the centroid of that category.

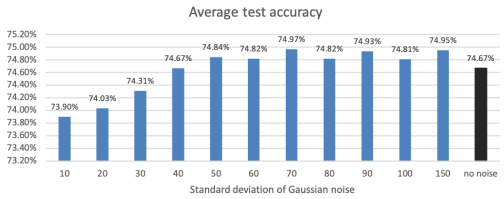
Table 4.1 shows the performance of the classifier after retraining with subgroups of the original data. We observe the following from the results: More training samples do not guarantee a higher accuracy (e.g., $S_{\mathrm{hc}}^{.25} \cup S_{\mathrm{li}}^{.25}$ vs. $S_{\mathrm{lc}}^{.50} \cup S_{\mathrm{hi}}^{.50}$ and $S_{\mathrm{hc}}^{.50} \cup S_{\mathrm{li}}^{.50}$ vs. $S_{\mathrm{lc}}^{.75} \cup S_{\mathrm{hi}}^{.75}$), as training samples may bring different effects with respect to classification on the test set.

High-confidence images (i.e., S_{hc}) are required for higher test accuracy, especially when a limited number of training samples are provided (i.e., in cases with $S_{\text{hc}}^{.25}$). Highly illusive images are misleading when the size of the training set is small (i.e., in cases with $S_{\text{hi}}^{.25}$); as the training set expands, however, such adverse images become valuable and lead to even higher accuracy than training with low-illusiveness ones (e.g., $S_{\text{hc}}^{.75} \cup S_{\text{li}}^{.75}$ vs. $S_{\text{hc}}^{.75} \cup S_{\text{hi}}^{.75}$). One possible explanation is that highly illusive images, as outliers, force the network to adjust for a lower loss. In this sense, the highly illusive images contain higher entropy (i.e., more information) than low-illusiveness images after a certain number of iterations. Classifiers trained with $S_{\text{hc}} \cup S_{\text{li}}$ are determined (smaller σ_A) and confident (higher P_c, P_i) about what they predict, regardless of whether they are right or wrong. By contrast, classifiers trained with $S_{\text{lc}} \cup S_{\text{hi}}$ are relatively hesitant (larger σ_A) in that the average probabilities for the output category (P_c, P_i) are lower; moreover, even if the prediction is wrong, they still assign a comparatively higher probability on the ground-truth category (P_g).

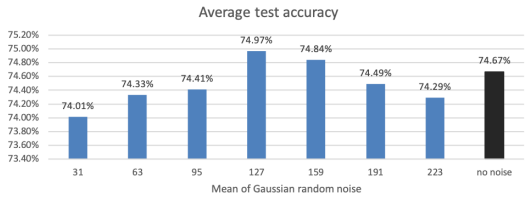
In summary, the results demonstrate that the performance of classifiers can depend heavily on the training data. Moreover, dataset bias occurs even within the same dataset because it is an intrinsic property of the learning scheme. In practice, our training set can be regarded as a subset of an ambient set. Unfortunately, we have little clue about the composition of our training data.

4.2 Training with Extra Category(ies) from Poorly Justified True Beliefs

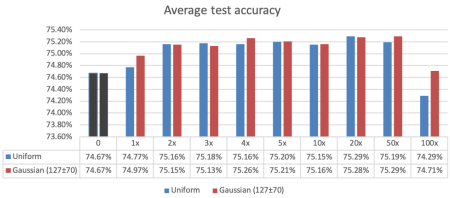
As discussed in Chapter 3.3.2, one shortcoming of an ordinary classifier is that it partitions the entire input space with only a limited number of subgroups. Adding extra categories to training data is the easiest way to change the topology of the learned space. In this section, we train classifiers with more categories than necessary and reveal potential benefits from such additional information. We explore three types of extra categories: random noise, auxiliary classes, and adversarial noise. According to the Euler diagram in Figure 3.5, one benefit from those extra categories is that they create room for true beliefs outside knowl-



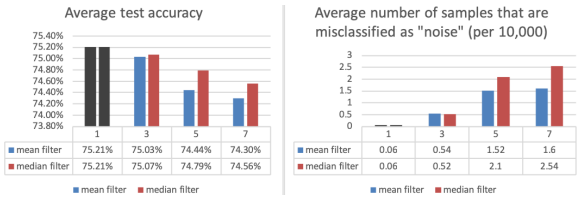
(a) Average accuracy vs. standard deviation. 1× noise samples centered at 127, scale 0



(b) Average accuracy vs. mean. 1× noise samples with std 70, scale 0



(c) Average accuracy vs. rate. Noise with mean 127, std 70, scale 0



(d) Average performance vs. scale. 5× noise samples with mean 127, std 70

Figure 4.1: Controlled experiments demonstrating the effectiveness of each factor in random noise.

edge. Such true beliefs are considered unjustified from a statistical viewpoint because the categories are not included in the test set. Another difference lies in that the underlying function to be approximated has been modified along with the increased number of categories.

4.2.1 Training with an Extra Class of Random Noise

We start with a naive cost-free category of random-noise samples. The use of random noise in training neural networks dates back to the 1990s. Researchers have injected noise into inputs and weights to improve the generalization [105], avoid local minima, and speed up back-propagation [106]. A recent study [98] used random perturbation to stabilize the networks. To the best of our knowledge, however, researchers have never treated random noise as independent training samples containing information that can be directly employed for training neural networks. In this experiment, we add an extra category of random noise as negative samples, retrain the network, and test the classifier on the original test set that does not contain samples of random noise. Surprisingly, with random noise we slightly improve the test accuracy at almost no extra cost except slightly longer training time.

We start testing with the most-common Gaussian random noise under four controlling parameters: mean, standard deviation, quantity, and blurring scale (i.e., the correlation among pixels). We design the experiments in a controlled manner to evaluate the effect of each factor. Figure 4.1a and 4.1b show that Gaussian random noise improves test accuracy as long as it is widely spread with a centered mean and sufficiently large standard deviation. As the standard deviation increases, Gaussian noise becomes more uniformly distributed. Figure 4.1c shows that test accuracy varies with different numbers of random-noise training samples. This finding suggests that random noise might contain a certain amount of information that could teach the classifier what "is not" a natural image. Such information, however, becomes saturated as the number of random-noise samples increases. With an excessive number (e.g., 100×) of noise samples, neural networks may become overwhelmed, spending a much longer time training but obtaining little enhancement.

We further repeat the previous experiments in 4.1 from Section 4.1 with the additional category of uniformly-distributed random noise, and report results at Table 4.2. Comparing the results from Tables 4.1 and 4.2, we observe the following: Training classifiers with random noise tends to increase test accuracy, yet such an effect is more obvious with fewer training samples. The introduced new risk of "misclassification as random noise" vanishes as the number of legitimate training images increases. In general, classifiers trained with random noise tend to be more determined (higher P_c, P_i, lower σ_A); surprisingly, for misclassified samples, the probability of the ground-truth category also increases (higher P_g).

The motivation of adding random noise is to change the topology of natural-image space by specifying the invalid zones that do not belong to the space. As a result, if we subtract the region of random noise from the classifier's input space, then the input space is no longer "dense" and has more holes. Geometrically, to avoid such holes, random noise tends to push the decision boundaries towards the centroid of each category. In addition, feeding neural networks with random-noise samples that are entirely unnatural can

Table 4.2: 50-time average performance of the classifier retrained with subgroups of CIFAR-10 training data and an extra category of uniformly distributed random noise. S_{c}^{p}: the top p percent of training samples selected with criteria c. The number of uniformly distributed noise samples is 5× the number of legitimate samples in each category. Evaluation metrics are average test accuracy (A), standard deviation of test accuracy (σ_A), average confidence of correctly classified samples (P_c), average illusiveness of misclassified samples (P_i), average probability of the ground-truth category for misclassified samples (P_g), and average number of test samples that are misclassified as "noise" (N).

Training set	A	σ_A	P_c	P_i	P_g	N
$S_{\mathrm{lc}}^{.25} \cup S_{\mathrm{li}}^{.25} \cup$ noise	30.82%	2.11%	28.31%	24.14%	15.43%	2.1
$S_{\mathrm{lc}}^{.25} \cup S_{\mathrm{hi}}^{.25} \cup$ noise	21.32%	0.91%	25.27%	23.40%	14.30%	1.08
$S_{\mathrm{hc}}^{.25} \cup S_{\mathrm{li}}^{.25} \cup$ noise	55.27%	0.86%	79.24%	58.60%	12.86%	5.36
$S_{\mathrm{hc}}^{.25} \cup S_{\mathrm{li}}^{.25} \cup$ noise	53.26%	1.06%	67.22%	44.81%	14.87%	2.78
$S_{\mathrm{lc}}^{.50} \cup S_{\mathrm{li}}^{.50} \cup$ noise	53.11%	1.36%	46.70%	37.26%	19.23%	0.9
$S_{\mathrm{lc}}^{.50} \cup S_{\mathrm{hi}}^{.50} \cup$ noise	40.82%	1.10%	35.05%	31.23%	18.55%	0.68
$S_{\mathrm{hc}}^{.50} \cup S_{\mathrm{li}}^{.50} \cup$ noise	66.37%	0.53%	88.56%	70.03%	11.61%	1.12
$S_{\mathrm{hc}}^{.50} \cup S_{\mathrm{hi}}^{.50} \cup$ noise	65.13%	0.65%	76.10%	50.47%	16.16%	0.72
$S_{\mathrm{lc}}^{.75} \cup S_{\mathrm{li}}^{.75} \cup$ noise	70.03%	0.81%	73.88%	53.26%	18.42%	0.14
$S_{\mathrm{lc}}^{.75} \cup S_{\mathrm{hi}}^{.75} \cup$ noise	61.95%	1.38%	54.95%	43.80%	20.69%	0.50
$S_{\mathrm{hc}}^{.75} \cup S_{\mathrm{li}}^{.75} \cup$ noise	71.05%	0.54%	92.43%	75.64%	10.54%	0.22
$S_{\mathrm{hc}}^{.75} \cup S_{\mathrm{hi}}^{.75} \cup$ noise	71.42%	0.55%	80.94%	55.75%	16.48%	0.34

help them learn what "is not" an object. However, as the blurring scale of Gaussian noise increases, more pixels are correlated, and samples become less unnatural. Therefore, as illustrated by Figure 4.1d, the effect is weakened by Gaussian noise with larger correlations among pixels.

At the end of this experiment, we report two failure cases that lead to interesting results. In one case, the test accuracy drops if we train the classifier with images of solid colors. The set of solid-color images can be interpreted as an embedded sub-hyperplane that 1) has the same or even lower dimension as decision boundaries, and 2) spreads across the entire input space. Classifying such sets in lower dimensions is difficult because their boundaries reside in even lower dimensions. For a better understanding, we may consider solving a curve-fitting problem with classification instead of regression. In the other case,

test accuracy also drops in cases of Gaussian random noise with various means, standard deviations, and scales. Such mixed samples come from several separated clusters in the input space. Since neural networks group samples into path-connected regions [44], mixed noise becomes a more complicated case.

4.2.2 Training with Out-of-target Auxiliary Classes

When target categories differ significantly in a dataset, extra information from random noise may become unnecessary, especially for deep neural networks with a larger capacity. In this experiment, we choose fine-grained datasets to benchmark the classification accuracy for deep neural networks with extra information from out-of-target categories. Differences among categories in the fine-grained datasets are more subtle as all categories belong to the same superclass, for instance, aircrafts [107], flowers [108], birds [109] and indoor scenes [110]. To simulate the "extra" knowledge without additional data collection, we split fine-grained datasets into halves and set the goal of classification to only the first half of the categories. Following the example provided by PyTorch[2], our ResNet-50 [111] models are trained for 90 epochs with various initial learning rates and common batch size of 128, momentum of 0.9, as well as weight decay of 10^{-4}. The learning rates also drop by a factor of 10 every 30 epochs. We train the models from both random initialization (i.e., from scratch) and the pretrained weights from ImageNet (i.e., fine-tuning).

Table 4.3 compares test accuracy on the first half of the categories in fine-grained datasets. In the controlled group (i.e., half), we train and test the classifiers with only the first half of the categories in that dataset. For the experiment group (full), we train the classifier with all categories in that dataset and test the classifier on only the first half of the categories. According to the results, training with more categories than necessary can lead to higher accuracy in general, especially for training from scratch or larger learning rates. As the number of categories increases, the underlying target function to be approximated

[2] `https://github.com/pytorch/examples/tree/master/imagenet`

Table 4.3: Top-1 test accuracy on the first half of the categories in fine-grained datasets. The classifiers are trained with either (a) the first half of categories (i.e., half) or (b) all categories (i.e., full). The predictions are limited to the first half only. Results show that training with more categories than necessary can lead to higher accuracy in general, especially for training from scratch.

Initialization	Random					
Initial learning rate	0.1		0.01		0.001	
Training set	half	full	half	full	half	full
Aircrafts [107]	2.28	**8.16**	7.56	**43.55**	5.04	**7.62**
Flowers [108]	5.81	**25.39**	55.55	**57.55**	18.66	**23.79**
MIT-67 [110]	15.63	**15.78**	33.39	**50.53**	17.38	**19.42**
Initialization	Fine-tuning from ImageNet					
Initial learning rate	0.1		0.01		0.001	
Training set	half	full	half	full	half	full
Aircrafts	4.32	**81.52**	7.74	**81.34**	63.59	**68.93**
Flowers	46.66	**94.91**	97.40	**97.84**	92.07	**92.83**
MIT-67	35.36	**60.09**	**86.34**	86.19	85.74	**86.34**

by the network also varies. Consequently, the classifier's beliefs are closer to truths when comparing the partitions in a fine-grained domain. In addition, more categories lead to better descriptors for differentiating the targets. The latter benefit, however, can be weakened in a fine-tuning scenario because the pre-trained network has already been exposed to many categories in advance. In practice, collecting data of additional relevant yet exterior categories and training with more categories than necessary has the potential to further push the state-of-the-art accuracy.

4.2.3 Adaptive Refinement with Adversarial Noise

When no additional data from relevant categories can be collected, we can further adopt the dynamic refinement paradigm in Algorithm 1, which adaptively correct the belief of a classifier on unseen noise samples during training. At each epoch, the adversarial examples are dynamically crafted by applying targeted adversarial attacks to randomly generated noise. In other words, the current classifier misclassifies those noise samples into one of the object categories with high confidence. We then correct such misclassification by explicitly label

Algorithm 1 Adaptive refinement with adversarial noise

Inputs:
M: total number of samples in the training set.
N: total number of categories in the training set, category index starts from 0.
$X = \{\mathbf{x}_i\}_{i=0}^{M-1}$: training samples including labels.
f_0: initialized classifier with $N + 1$ output categories.
$p_Z(z)$: distribution of random-noise samples.
for epoch $t = 0$ **to** $T - 1$ **do**
  if $t > 0$ **then**
    Sample a batch of noise samples $\{z_t\} \sim p_Z(z)$;
    Generate adversarial examples $\tilde{z}_t$ with high confidence using f_t and z_t;
    Relabel the adversarial examples as category N;
    Refine the model with $\tilde{z}_t$
  end if
  Train 1 epoch with original training samples X to obtain f_{t+1};
end for
Output: f_T: trained classifier

Table 4.4: Top-1 test accuracy on fine-grained datasets with adversarial refinement. The classifiers are fine-tuned from ImageNet-pretrained ResNet-50. Adversarial refinement can marginally increase the test accuracy.

Dataset	Baseline	Refined
Aircrafts [107]	85.30	**85.54**
Flowers [108]	96.13	**97.01**
CUB [109]	79.08	**79.91**

those samples as random noise (i.e., a new category) and feed them back to the network as training samples. By intuition, the refinement process erases the most-confusing misclassified points from the subsets. As a result, the strategy increases the classification accuracy when fine-tuning pretrained ImageNet models on fine-grained datasets. Table 4.4 reports the accuracy enhancement from adversarial refinement. All configurations of the training remain the same as those in Section 4.2.2, except for a smaller initial learning rate of 0.01 and batch size of 48.

Although training with a background class is common, to the best of our knowledge, researchers have rarely treated random noise as a collection of independent training samples containing information that can be directly employed for training neural networks, even though most of the input space is covered by noise. Compared to the background

category, which consists of a limited number of images sampled from a particular dataset, our random-noise category contains an unlimited number of samples that can be generated on the fly. Different from adversarial training [24], the goal of which is to enhance local robustness against adversarial attacks on the classifier (usually with a slight sacrifice on the test accuracy), the goal of adversarial refinement is to increase test accuracy on clean samples. Moreover, each adversarial example in adversarial training is associated with two target categories (i.e., the misclassified and the ground-truth category), whereas in adversarial refinement, each adversarial example only refines for one target category without occupying space from other target categories. Our adversarial refinement is also conceptually connected with introspective networks [112], whose goal is to deliver a generator with better imagination capability under the support of the network's differentiation. On the contrary, our adversarial refinement aims at better differentiation capability with the help of its imagination.

4.3 Distillation from False Beliefs

It is often believed that a clear distinction lies between benign and adversarial examples. In the manifold assumption, benign images are considered on-the-manifold, whereas adversarial examples fall off the natural-image manifold. In this section, we argue that the two groups are not irreconcilable in that adversarial examples can positively impact classification on benign images in a manner called belief distillation.

4.3.1 The Detector's Dilemma

Let us begin by considering the following paradoxes for an adversarial detector, the duty of which is to judge whether an input is adversarial or not.

The detector's dilemma: If we perturb an adversarial example back to its original (i.e., correct) category with another targeted attack, should the re-perturbed image be considered as benign or adversarial? Similarly, what about perturbing a misclassified benign image

Table 4.5: Validation of the detector's dilemma and the manifold assumption. Acc(val): accuracy on the validation/test set, Acc(adv): accuracy on adversarial examples derived from the validation/test set. Acc(adv-pert)/Acc(mis-pert): accuracy on reperturbed adversarial examples and perturbed misclassified validation samples, respectively. Manifold(adv-pert)/manifold(mis-pert): percentage of on-the-manifold according to the detector for reperturbed adversarial examples and perturbed misclassified samples, respectively.

Dataset	**Acc(val)**	**Acc(adv)**	**Acc (adv-pert)**	**Manifold (adv-pert)**	**Acc (mis-pert)**	**Manifold (mis-pert)**
CIFAR-10	73.87%	13.05%	76.37%	0%	19.19%	0%
Tiny-ImageNet	59.20%	8.68%	95.00%	0%	95.60%	0%
ImageNet	76.15%	2.39%	99.32%	0%	99.33%	0%

to its ground-truth category?

For both scenarios, the obtained images can be regarded as adversarial because their outputs differ from previous ones. On the other hand, they should not be treated as adversarial examples in that they are correctly classified eventually. Those perturbations should be considered as "beneficial" at best.

To verify the detector's dilemma, we conduct experiments on CIFAR-10 [104], Tiny-ImageNet [113] and ImageNet [66], in which base classifiers are trained with default settings and hyper-parameters.[3] For ImageNet, we simply adopted the pretrained ResNet-50 model[4]. After obtaining the base classifiers, we generated adversarial examples for both the training and validation sets using projected gradient descent (PGD) attack [31] provided by IBM Adversarial-Robustness-Toolbox [39]. The misclassified samples in the validation set were also selected and stored. We then applied targeted PGD attacks to (1) naturally misclassified validation samples and (2) adversarial validation examples using their ground-truth categories as targets, thus obtaining (1) perturbed misclassified samples and (2) reperturbed adversarial examples, respectively. Meanwhile, we trained a detector (i.e., a binary classifier) using the original training set and adversarial examples generated

[3] `https://github.com/IBM/adversarial-robustness-toolbox/blob/master/examples/adversarial_training_cifar10.py` and `https://github.com/pytorch/examples/blob/master/imagenet/main.py`

[4] `https://pytorch.org/docs/stable/torchvision/models.html`

Table 4.6: Test accuracy with belief distillation: Training classifiers with adversarial examples and incorrect labels generated from a pretrained network.

Dataset	Chance	Belief distillation	Baseline
CIFAR-10	10%	23.17%	73.64%
Tiny-ImageNet	0.5%	32.20%	59.20%
ImageNet	0.1%	24.42%	76.15%

from the training set. Finally, all validation sets were evaluated using the base classifier and the detector.

Table 4.5 reports classification accuracy and percentage of being on-the-manifold. According to the results, the percentage of being on-the-manifold is always zero once the input is adversarially perturbed, which indicates that the detector essentially learns noisy fingerprints. Moreover, further experiments show that if we smooth the adversarial examples, the percentage will increase. The noisy fingerprints might be a side-effect of the gradient-based attacks, which suggest smoothing as a defense. In section 4.4, we apply various smoothing techniques at test time as a defense against adversarial attacks. Another observation is that as resolution increases, it becomes easier to attack an image and manipulate the desired predicted labels. Such a phenomenon agrees with the properties of the proposed hypothetical image-space model described in Section 3.2.1. With a sparser natural-image space in a higher dimension, more uncertain areas are required to connect training samples.

4.3.2 Belief Distillation

To demonstrate that benign and adversarial examples are not irreconcilable, we illustrate the belief distillation phenomenon by retraining the classifiers on adversarial examples and their incorrect labels that are derived from the base classifier. Table 4.6 reports the classification accuracy on the original validation set of CIFAR-10, Tiny-ImageNet and ImageNet. Although the accuracy is far away from the ones with the base classifier, it is much better than chance. On MNIST [114], one can even train a classifier using pure adversarial examples completely generated from random noise. The test accuracy with the classifier trained

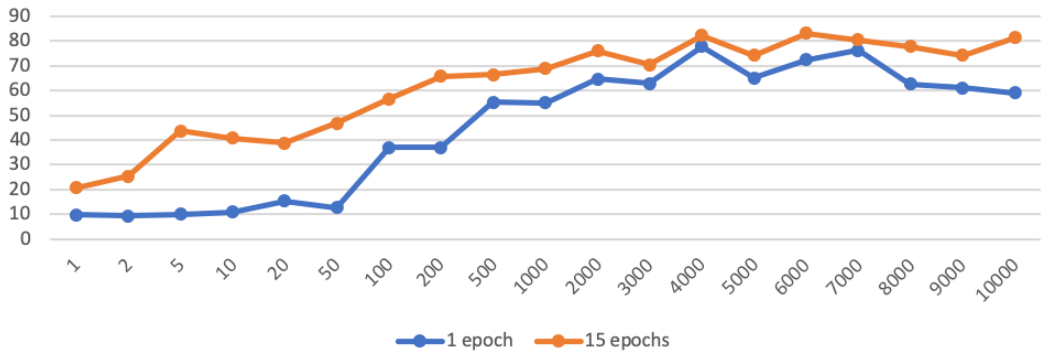


Figure 4.2: Training an MNIST classifier with pure adversarial examples generated from random noise. Horizontal axis: number of training samples per category; Vertical axis: test accuracy.

on pure adversarial noise is shown at Figure 4.2. The adversarial noise samples for belief distillation is generated by applying targeted adversarial attacks to base classifier and updating the random-noise inputs following the gradient. The classifiers we obtained in those experiments can be regarded as a student network of the base classifiers, whose belief is "distilled" via the adversarial examples, even though the belief is false. Both the teacher and the student agree with the adversarial training samples and their incorrect labels at high confidence, and such an agreed partition can somewhat be generalized to clean samples.

The methodology of belief distillation has multiple names in literature, such as adversarial belief matching [115] and dreaming to distill [116]. We will analyze those concurrent work as case studies. Our topological view of knowledge provides a complementary explanation of the non-robust features discovered in [50] and the adversarial belief matching proposed in [115] for zero-shot knowledge transfer. In adversarial belief matching, the student model is trained with pseudo data on which the student poorly matches the teacher. Instead of utilizing the teacher network's imagination, the pseudo data is produced by a separate generator network. As belief distillation can help preserve the partition from a classifier, it is tempting to apply the methodology in a continual learning environment, in which the goal is to preserve the partition for categories in previous tasks. Successful applications in continual learning are reported in [116]. According to the viewpoint of our paper,

the success may depend on two conditions. One is that the classifier needs to be sufficiently knowledgeable (e.g., w/ a large number of categories) after the first task so that false beliefs are in a stronger correlation with the true ones. As for the other condition, the required descriptors for differentiating categories in those tasks should be similar. Therefore in [116], ImageNet is chosen as the initial task to ensure that the classifier is sufficiently knowledgeable for belief distillation. Meanwhile, subsequent tasks are built from fine-grained datasets that require similar descriptors as ImageNet. Transfer learning with fixed ImageNet feature extractor achieves high classification accuracy on those fine-grained datasets [67].

When it comes to the limitation of belief distillation, it is worth pointing out that the success of belief distillation relies heavily on the implicit yet necessary assumption that all inputs are expected to be classified as one of the categories on which the teacher classifier have been trained. The student classifier then learns a posterior distribution given that inputs must belong to one of the categories. The correct predictions from the student classifier, however, do not ensure that the classifier has learned useful features of certain objects. When we replace the adversarial noise with the original training samples completely for some categories in the MNIST experiment shown in Figure 4.2, the classifier fails to learn the categories composed of adversarial noise. As suggested by Section 4.2.1, a barrier seems to exist between natural images and unnatural adversarial noise. For similar reasons, memory replay with belief distillation fails on continual learning benchmarks [70], such as split MNIST [117].

4.4 Adaptive View of Adversarial Robustness

Motivated by experiment results in Section 4.3.1 that the percentage of being on-the-manifold according to the adversarial detector increases as the re-perturbed images are smoothed, we apply various smoothing techniques in this section as a test-time defense against adversarial attacks. Instead of proving the superiority of a particular method, we present an adaptive view of adversarial robustness with the discovered non-monotonic

relation between adversarial attacks and smoothing defenses. Meanwhile, the test-time smoothing methods discussed in this section can still complement other defense schemes, such as adversarial training [24].

In general, the experiments are designed based on the following principles. We limit our scope to white-box untargeted attacks, which is the most common type in literature. Compared with the feature denoising in [43], all our smoothing defenses are performed at test time. We assume that the neural-network classifier has already been shipped and deployed, or it might not be feasible to retrain with adversarial examples. Contrary to existing work, which often compares methods at the dataset level with static configurations (e.g., a few sets of fixed parameters), we thoroughly investigate the behavior of test-time defenses at multiple levels and varying strengths. Smoothing methods are suitable for illustration because their strength can be naturally measured using the number of iterations or radius of kernels.

4.4.1 Test-time Defense Scheme

We begin by elaborating on a general test-time defense scheme. Let $f : X \to Y$ denote a pretrained classifier that maps an image $x \in X$ to its label $y \in Y$. An adversarial attack $a : X \to X$ then maps a legitimate image x to an adversarial example $\hat{x}$ under certain constraints (e.g., on L_p distance) such that $f(x) \neq f(\hat{x}) = f(a(x))$. To defend adversarial attacks at test time, an ideal solution would be applying the inverse mapping $a^{-1} : \hat{x} \mapsto x$. In reality, however, we have to find a defense h, which is an alternative approximation of g^{-1} with the hope that $f(x) = f(h(\hat{x}))$ can be satisfied. In addition, a defense h is more desirable for deployment if it brings less distortion to legitimate images x, keeping $f(x) = f(h(x))$. To achieve that, we may also introduce a detector (as a part of h) that distinguishes adversarial examples from legitimate ones at the first stage of the defense. Once an input is considered legitimate, no further defense is required. In this work, we apply smoothing techniques as the alternative approximation (h) of the inverse mapping (a^{-1}) of the attack. Theoretically,

the smoothing defense only works when outputs from an adversarial attack (a) are "noisy," which implies $a \approx h^{-1}$ when composite with f (i.e., $f \circ (h \circ a) \approx f$).

4.4.2 Smoothing Techniques

We implement a test-time defense pipeline that can smooth both the original inputs and intermediate features from any specified layer(s) of the neural-classifier. The smoothing techniques that we experiment with can be categorized into three groups: common, edge-preserving, and advanced. The common group includes mean, median, and Gaussian filters, which are most commonly used in image processing. Edge-preserving smoothing algorithms include anisotropic diffusion and bilateral filter. More advanced smoothing techniques include non-local means and modified curvature motion. We will explain the algorithms concisely in the following paragraphs.

Mean, median, and Gaussian filters: These filters are widely applied in image processing. Despite the simple forms, they do not necessarily perform the worst in defending adversarial examples.

Anisotropic diffusion [118]: The Perona-Malik anisotropic diffusion aims at reducing image noise without removing important edges by assigning lower diffusion coefficients for edge pixels (which have larger gradient norm). During iterations, the image is updated according to the formula below.

$$I_t = \operatorname{div}(c(x, y, t)\nabla I) = \nabla c \cdot \nabla I + c(x, y, t)\Delta I \tag{4.1}$$

in which div denotes the divergence operator, Δ denotes the Laplacian, and ∇ denotes the gradient. The diffusion coefficient is defined either as $c(\|\nabla I\|) = e^{-(\|\nabla I\|/K)^2}$ or $c(\|\nabla I\|) = \frac{1}{1+(\|\nabla I\|/K)^2}$.

Bilateral filter [119]: A bilateral filter is a non-linear edge-preserving filter that computes the filtered value of a pixel p using weighted average of its neighborhood S. The

filter can be expressed with the following formula [120].

$$\mathrm{BF}[I](p) = \frac{1}{W(p)} \sum_{q \in S} G_{\sigma_s}(\|p - q\|)\, G_{\sigma_r}(|I(p) - I(q)|)\, I(q) \tag{4.2}$$

in which $W(p) = \sum_{q \in S} G_{\sigma_s}(\|p-q\|)\, G_{\sigma_r}(|I(p)-I(q)|)$ is the normalization term. G_{σ_s} and G_{σ_r} are weight functions (e.g., Gaussian) for space and range, respectively. Edges are preserved because pixels that fall on different sides of the edge will have lower weights for range.

Non-local means [121]: The non-local mean algorithm takes a more general form in which the output value of a pixel i is computed as a average of all pixels in the image I, weighted by a similarity $w(i, j)$ which is measured as a decreasing function of the weighted Euclidean distance to that pixel. For a discrete image $v = \{v(i) \mid i \in I\}$, the filtered value for a pixel i is computed as follows.

$$\mathrm{NL}[v](i) = \sum_{j \in I} w(i, j) v(i) \tag{4.3}$$

in which the weights $w(i, j) = \frac{1}{Z(i)} e^{-\frac{\|v(\mathcal{N}_i) - v(\mathcal{N}_j)\|^2_{2,a}}{h^2}}$. In the formula, $\mathcal{N}_k$ denotes a square neighborhood of fixed size centered at a pixel k, and a is the standard deviation of the Gaussian kernel. The normalizing constant is computed as $Z(i) = \sum_j e^{-\frac{\|v(\mathcal{N}_i) - v(\mathcal{N}_j)\|^2_{2,a}}{h^2}}$.

Modified curvature motion [122]: As most smoothing techniques are initially designed for gray-scale images, generalizing them to multi-channel color images and feature maps might be less natural, and sometimes there may exist multiple ways for the generalization. Instead of splitting a color image into separate channels, we can treat it as a surface $(x, y, R(x, y), G(x, y), B(x, y)) \subset \mathbb{R}^5$. Following the geometric property that smoother surfaces have smaller areas (or volumes), we can then iteratively smooth it with a general curvature motion method:

$$I_t = \frac{k^{-2}\nabla^2 I + (I_y^2 + I_z^2) I_{xx} + (I_x^2 + I_z^2) I_{yy} + (I_x^2 + I_y^2) I_{zz} - 2(I_x I_y I_{xy} + I_x I_z I_{xz} + I_y I_z I_{yz})}{(k^{-2} + \|\nabla I\|^2)^2} \tag{4.4}$$

where k is a scaling factor. As k becomes larger, the algorithm transits from isotropic to a more edge-preserving diffusion. Such a formulation can be easily and naturally extended to feature maps with more channels along the z axis.

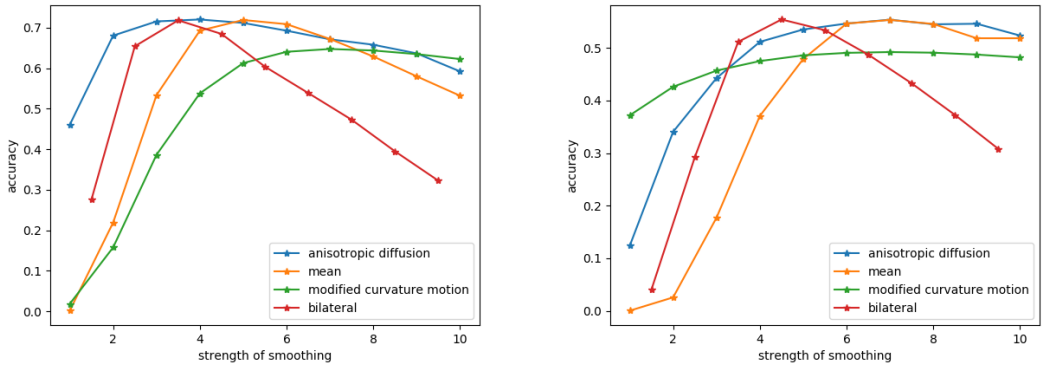
4.4.3 Non-monotonic Relation between Attacks and Defenses

We now present our experimental details of test-time smoothing against adversarial attacks. In order to prepare the test set, we first select from the ImageNet [66] validation set all images (39,156 in total) that are correctly classified by a pretrained ResNet-152 [111]. Then we generate and store adversarial examples using attacks that are provided by Foolbox [123] and ART [39]. The white-box untargeted attacks include Projected Gradient Descent (PGD) [31], Deep Fool[29], Saliency Map[28], Newton Fool [124], and salt-and-pepper noise. For strong attacks (e.g., PGD) that cannot be mitigated by quantization, we store the adversarial examples in *jpeg* format; for other attacks that require adversarial examples in floating-point accuracy, we store their results in *pkl* files.

Performance of various smoothing techniques on defending fixed adversarial attacks

We conduct a set of controlled experiments that show the performance of various smoothing techniques on defending a fixed attack. Two sets of PGD attacks, with maximum perturbation $\epsilon = 0.01$ (i.e., imperceptible to humans) and $\epsilon = 0.05$ (i.e., similar scale as in [43]), are chosen as a baseline because (1) PGD attack is one of the strongest attacks and (2) adversarial examples from PGD attack cannot be defended by simple quantization. Following the default settings in ART [39], 20 iterations of PGD are performed. After obtaining the perturbed images, we tweak the parameters in each smoothing method to pursue the optimal ones that lead to the highest classification accuracy over all perturbed images. If a method contains multiple parameters, we tweak them one after another and naively apply the optimal parameter values obtained from the previous exploration.

Figure 4.3 shows the classification accuracy on ImageNet validation set as the strength



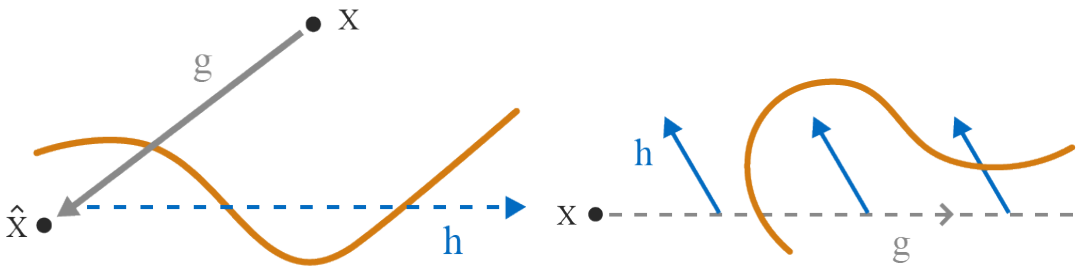
(a) PGD ($\epsilon = 0.01$, 20 iterations) (b) PGD ($\epsilon = 0.05$, 20 iterations)

Figure 4.3: Change of classification accuracy on ImageNet validation set (vertical) along with the strength of smoothing defense (horizontal). The strength of smoothing method is measured by the most sensitive parameter: number of iterations for anisotropic diffusion and modified curvature motion, size of the kernel for mean filter, and radius of the neighborhood for bilateral filter.

of smoothing defenses varies. The most sensitive parameter measures the strength, that is, *number of iterations* for iterative methods such as anisotropic diffusion and modified curvature motion, *size of the kernel* for mean filters, and *diameter of the neighborhood* for bilateral filters. We only present results from four selected methods because the rest of them lead to much lower (i.e., 20-30% less) classification accuracy. Henceforth, we will focus on these four methods in subsequent experiments. The curves in Figure 4.3 share a similar concave shape, which might suggest a geometric relation between adversarial attack a and test-time defense h. As illustrated by Figure 4.4a, the test-time defense should not travel too far along the "detour." In the following subsection, we further study the non-monotonic effect from the attackers' perspective.

Performance of a fixed defense under attacks with varying number of iterations

We continue our controlled experiments by setting the parameters of each smoothing defense to the optimal values obtained in the previous experiments and varying the strength (i.e., number of iterations) of PGD attacks. Figure 4.5 presents the classification accuracy on ImageNet validation set as the number of attack iteration increases from 1 to 100.



(a) fixed attack a, varying defense h (b) varying attack a, fixed defense h

Figure 4.4: Simplified illustration on the "detour" effect between adversarial attack a and test-time defense h

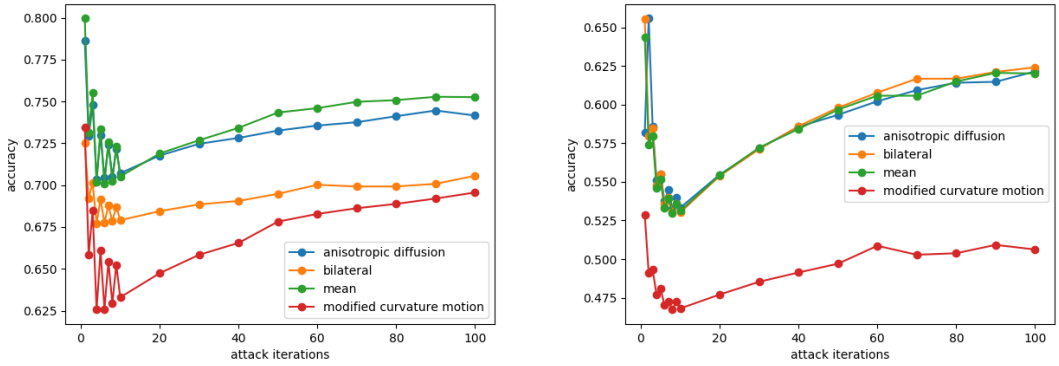
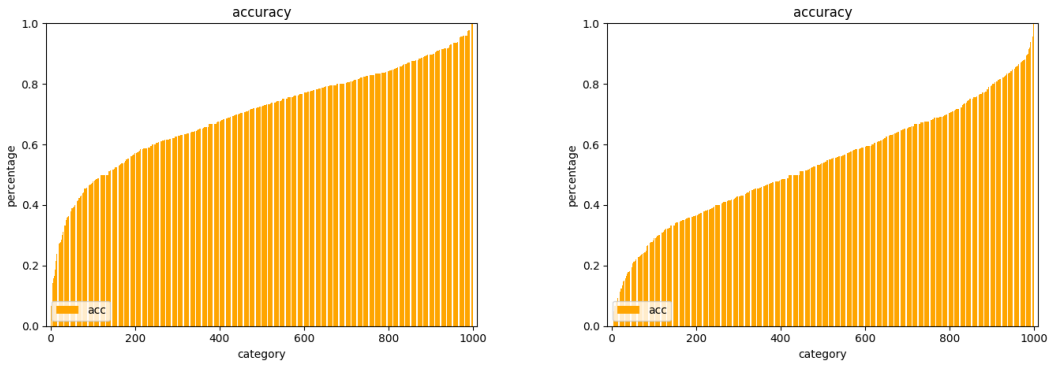


Figure 4.5: Change of classification accuracy on ImageNet validation set (vertical) along with the number of iterations in PGD attack (horizontal). The bump at iteration = 50 corresponds to a switch from the ImageNet validation set to a subset of 5,000 images to reduce computation time.

Surprisingly, the accuracy first drops but then rebounds as the number of attack iterations keeps increasing. Such performance might seem contradictory to previous work as we used to believe that more iterations leads to stronger attacks, especially for defenses that involve adversarial training. For test-time defenses, however, the convex curves may reflect the actual non-monotonic relation between attacks a and defenses h, as illustrated in Figure 4.4b. In contrast, when no smoothing defense is performed, the classification accuracy keeps dropping and stabilizes at a low level.



(a) PGD, $\epsilon = 0.01$ (b) PGD, $\epsilon = 0.05$

Figure 4.6: Distribution of categorical accuracy on adversarial examples in increasing order. (a): Results on adversarial examples that are generated from PGD ($\epsilon = 0.01$), with anisotropic diffusion as defense. (b): same as (a) but the adversarial examples are generated from PGD ($\epsilon = 0.05$).

4.4.4 Variance of Robustness

Variance of robustness among categories

During the experiments, we noticed that the variance of classification accuracy for each category was quite large. For illustration purposes, we take PGD attack and anisotropic diffusion as an example. Figure 4.6 shows the sorted accuracy from ImageNet categories. The lowest categorical accuracy stays below 20%, whereas the highest reaches almost 100%. A similar distribution of categorical accuracy is observed from other attack-defense pairs. The observation motivates us to investigate properties that are correlated to this large variance. Unfortunately, correlations with categorical accuracy are not observed among seemingly plausible factors, including the categorical confidence on unattacked samples and probability assigned on the ground-truth category in adversarial examples. Details of the investigation are available in Appendix B.

The huge variance on categorical accuracy also leads us to a question: is it possible to select a relatively large subset of test samples on which a designated method works the best? The task turns out to be easy. For each smoothing technique, we sort the test samples that are correctly classified according to their prediction confidence. Then, we can select

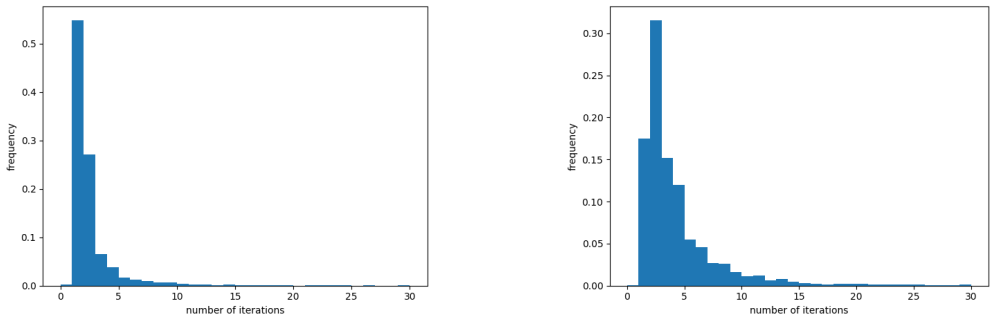
Table 4.7: Classification accuracy on "optimal" subsets consisting of adversarial examples generated from PGD attack ($\epsilon = 0.01$). Accuracies can be inflated to 100% on a dataset with $> 10^4$ samples.

Defense / "Optimal" subset	Anisotropic diffusion	Bilateral	Mean
Anisotropic diffusion	100.0%	88.51%	93.79%
Bilateral	92.79%	100%	93.81%
Mean	93.62%	89.31%	100%

a relatively large (with more than 20,000 samples) "optimal" subset with high prediction confidence. The performance on those optimal subsets are shown in Table 4.7. Due to the nature of large variance, the choice of the testing set has a high impact on the final result. For example, the subset at the first row is selected based on anisotropic diffusion. Therefore, anisotropic diffusion achieves 100% accuracy, while bilateral filters only achieve less than 90%. The possibility and simplicity of inflating the test accuracy (even to 100%) raise concerns on the validity of results from small-scale or private datasets in previous studies and calls for rigorous evaluation metrics at finer levels and larger scales.

Variance of required defense for each sample

Both non-monotonic relations in section 4.4.3 and large variance in section 4.4.4 suggest the idea of an adaptive version of the test-time smoothing defense, which is favorable for iterative methods. Specifically, the optimal iteration number or termination criterion varies sample by sample. In order to demonstrate the potential advantage of the adaptive method, we compute the minimum number of iterations required for defending an adversarial example. Figure 4.7 shows the histograms of minimum iterations required with anisotropic diffusion under two sets of PGD attacks. In addition, the upper bound of the minimum iteration number is set to 30. In other words, if an adversarial sample remains misclassified throughout 30 smoothing iterations, we consider it as undefendable. We then compute an upper-bound accuracy for the defense by taking account results from all iterations. Compared with the result from a fixed iteration number over the whole dataset, our simulation of



(a) PGD ($\epsilon = 0.01$) (b) PGD ($\epsilon = 0.05$)

Figure 4.7: Histogram of the minimum number of iterations required for defending adversarial examples with anisotropic diffusion. Horizontal axis: number of iterations; vertical axis: proportion of the training samples.

the "adaptive method" enhance the accuracy from 72.2% to 83.6% on adversarial examples generated by PGD ($\epsilon = 0.01$) and from 55.5% to 70.1% on adversarial examples generated by PGD ($\epsilon = 0.05$).

Theoretically, test-time smoothing only defends high-frequency perturbations that are noisy. In practice, there exist samples that are consistently misclassified even though they are not adversarially perturbed with high-frequency noise. In the next section, we study those misclassified illusive samples which do not contain high-frequency noise component.

4.5 Verifying Geometric Causes of Adversarial Examples

In this section, we verify (or partially verify) three geometric causes[5] of adversarial examples under the guidance of the hypothetical image-space model. The section is composed of several subsections, each containing a hypothesis on the causes of adversarial examples and the verification process. The subsections include the inspiration of the hypothesis, the experiment design, and the results of the controlled experiments. It should be emphasized that this section's focus is by no means proposing state-of-the-art attacks or defenses. Moreover, we may lose contrast in the results under strong attacks. To provide readers more apparent comparisons as well as illustrating the easiness of attacking classifiers, we

[5]For completeness, verification of the statistical causes is attached in Appendix E.1.

adopt a fast weak and untargeted attack, namely Fast Gradient Sign Method (FGSM) [24], provided by IBM-ART [39].

4.5.1 Path-connected regions from classifiers

Hypothesis A *Adversarial examples exist at uncertain "bridges" created by the classifier for connecting samples of the same category in a path-connected manner.*

Reasoning: As pointed out in [44], neural-network classifiers tend to partition the input space into path-connected regions. Given that classifiers have limited, finite capability of approximating the partition from truths, there will often be samples that are consistently misclassified during training, especially in early epochs. Henceforth, we will refer to those as illusive samples. Such illusive samples agree with the existence of natural adversarial examples [125] as they both belong to false beliefs that do not match the truths from the oracle. To connect those separated samples with path-connected regions, a network may need to deform its decision boundaries, creating somewhat arbitrary bridges that pass through uncertain regions in between. A toy example that illustrates the bridges is presented in Appendix E.3. We guess that if we exclude those illusive samples during training, the obtained classifier may become more robust because less uncertain areas are required to establish the connection.

Design principles: We train classifiers with the same model and configurations on the following different training sets:

- the entire original training set (control);
- original training set without illusive samples (experimental);
- original training set after randomly removing the same number of samples as the number of illusive samples in the experimental group (control).

To identify illusive samples, we train multiple shallow classifiers and compute training statistics, such as counting the times that a particular training sample has been correctly

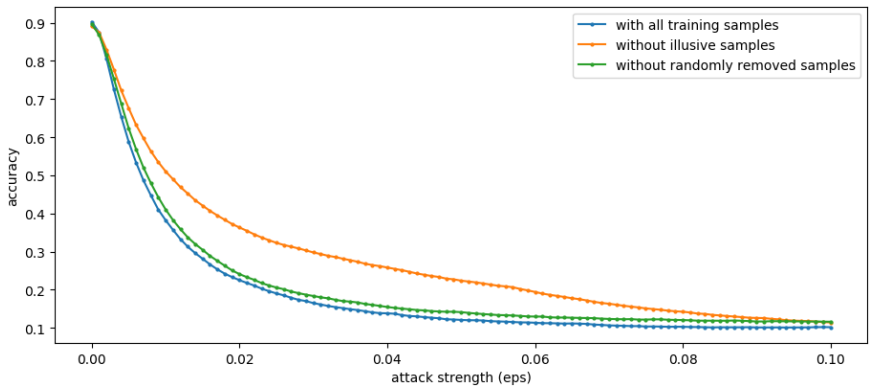


Figure 4.8: Training without consistently-misclassified illusive samples help enhance robustness against adversarial attacks.

classified. Shallow classifiers are preferred for this procedure because deep ones with huge model capacity may easily classify most (if not all) training samples correctly.

Experiment: The illusive samples are selected based on the training statistics obtained from training 10 randomly-initialized shallow classifiers[6] on CIFAR10 [104] for 25 epochs. If all ten classifiers always misclassify a training sample, we consider it an illusive sample. We then train a deeper classifier on CIFAR10 using the CleverHans toolbox [38] on the three datasets listed above. Figure 4.8 shows the accuracy of adversarial test samples for the three classifiers trained with different training sets. When no defense is performed, training without illusive samples leads to higher accuracy on adversarial examples with a slight accuracy drop on clean test samples. Therefore, the adversarial robustness of a classifier is related to the illusive samples in the training set. In addition, we also notice that training without them alleviates the strong incorrect momentum, which results in accuracy drops at late epochs, of which the details are provided in Appendix E.2.

Self-reflection: We must admit that to justify the hypothesis adequately and rigorously is exceptionally challenging: one shall define, locate, or even visualize the uncertain "bridges" in an exceedingly high-dimensional image space. Our experiment, alternatively, presents an indirect verification via results that are consistent with the hypothesis. Another clarification is that path-connectedness is not a negative characteristic at all times: it some-

[6] `https://keras.io/examples/cifar10_cnn/`

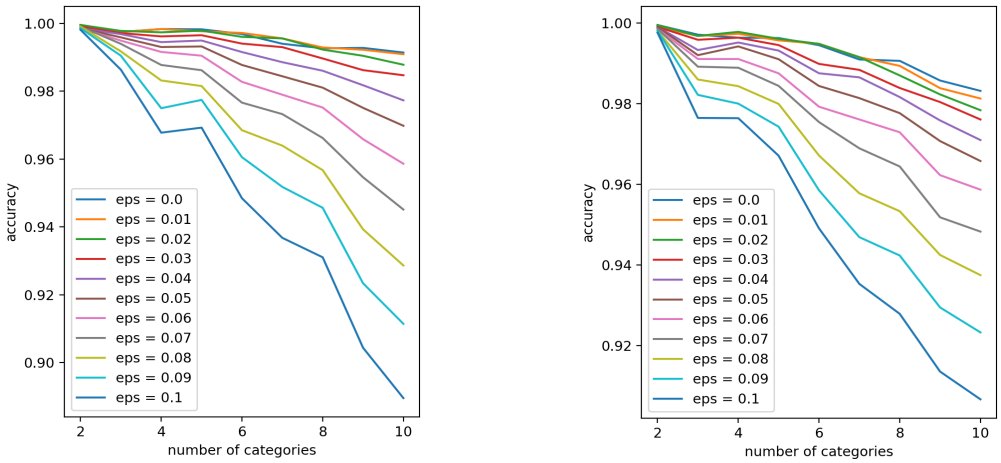
how fulfills generalization to unseen test samples. It is, however, more desirable should the classifier be capable of preserving only necessary connections. We have attempted to allow multiple clusters and distributions for each category so that unnecessary "bridges" can be avoided in the partition. Unfortunately, almost all samples in a category are assigned to the same cluster by the classifier. A key part of future studies is to break the constraint on path-connected regions.

4.5.2 Excessive number of target categories

Hypothesis B *Classifiers trained for fewer target categories tend to be more robust than those trained for more target categories.*

Reasoning: As neural networks tend to partition an image space into path-connected regions [44], the categories are expected to intertwine with each other as the total number increases. In addition, given that classifiers tend to place all input samples (even randomly sampled ones in the input space) close to the boundaries [45], more categories would give rise to more possibilities for attacks.

Design principles: For demonstrating the impact from the number of target categories, we train 9 classifiers (offline) on subsets of categories from MNIST [114] by gradually adding the categories in sequential order (i.e., additive mode), starting from the first two categories $\{0, 1\}$ to all 10 categories $\{0, 1, \cdots, 9\}$. To rule out the potentially dominant impact from the change of the total number of training samples, we also train those 9 classifiers by keeping (1) the total number of training samples constant and (2) the number of samples for each category balanced (i.e., constant mode). For all classifiers, the number of neurons in the last layer is equal to the number of target categories. It is desirable to tune the initial accuracy on clean test samples (i.e., $\epsilon = 0$) so that they are at a similar level for all classifiers trained on a various number of target categories; thus the accuracy drops under adversarial attack shall be ascribed to the loss of robustness. One reason for using MNIST



(a) additive mode: including all available training samples

(b) constant mode: 10,000 training samples (balanced)

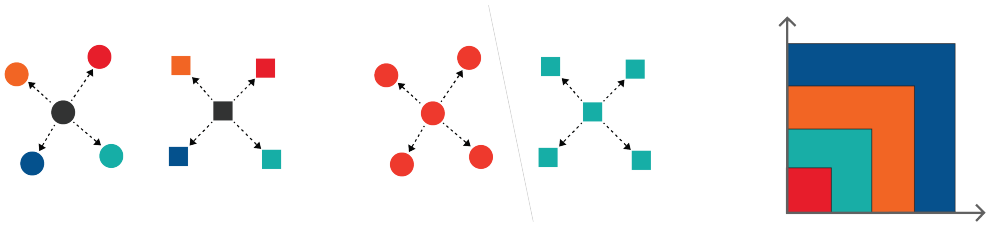
Figure 4.9: Robustness of classifiers decreases as the number of target categories increases. ϵ (eps): the strength of attacks.

in our experiment is that such a goal is much easier to reach using this dataset.[7] For most public datasets, adding target categories will decrease the accuracy as the task becomes noticeably harder. We then evaluate the robustness of those classifiers using adversarial examples from FGSM attack at varying strength. When evaluating a particular classifier, test samples from only the trained categories are involved.

Experiment: We adopt the model architecture from the official MNIST example from PyTorch,[8] and adjust the output shape of the last layer to match the number of categories. Following the design principles, we train two groups of classifiers in both the additive and constant mode. For the constant mode, we fix the number of total training samples to 10,000. The robustness of the classifiers is then examined via FGSM attacks with various strengths. Figure 4.9 verifies the loss of robustness as the number of target categories increases, regardless of the number of total training samples.

[7] MNIST is perhaps one of the few real datasets on which a modern classifier can achieve similar accuracy as on its subsets of categories.

[8] `https://github.com/pytorch/examples/tree/master/mnist`



(a) style classes (b) content classes (c) band classes

Figure 4.10: Partition of an input space with different categories

4.5.3 Geometry of categories (entropy of the category distribution)

Hypothesis C *The adversarial robustness of a classifier depends on the geometry of the input space (i.e., the entropy of the distribution of categories). Tentatively, the robustness tends to be positively correlated to the ratio of average inter-class distance d_{inter} to intra-class distance d_{intra} among samples.*

Reasoning: From a geometric perspective, the training samples can be regarded as placeholders (i.e., "anchor points") for their categories. During training, the classifier manages to deform its boundary such that classification errors on those "anchor points" are minimized. Depending on the intrinsic distribution of those "anchor points," the learned classifier can exhibit different adversarial robustness even with exact model architecture and training configurations.

Design principles: To verify the hypothesis, we prepare three datasets collected from the same image space (i.e., with identical input dimension) with same number of categories and samples yet with significantly different geometry. The datasets are constructed by classes which partition the input spaces in distinctive manners as illustrated in Figure 4.10:

- *Style classes*: The style classes (Figure 4.10a) are borrowed from our preset classification benchmark[9] of which the goal is to recognize global photo-editing styles. Starting from a base image (i.e., the dark-gray one in the center) in MIT-Adobe FiveK dataset [87], we obtain 10 stylized version of the same content by applying artistic

[9]More elaboration on the benchmark is provided in Chapter 5.2.2. Here we present necessary description of the dataset for coherent purpose.

presets using Adobe Camera Raw, resulting in a total of 11 style categories (including the original ones). Due to the control on image content, the intra-class distances d_{intra} among samples are often larger than the inter-class distances d_{inter}.

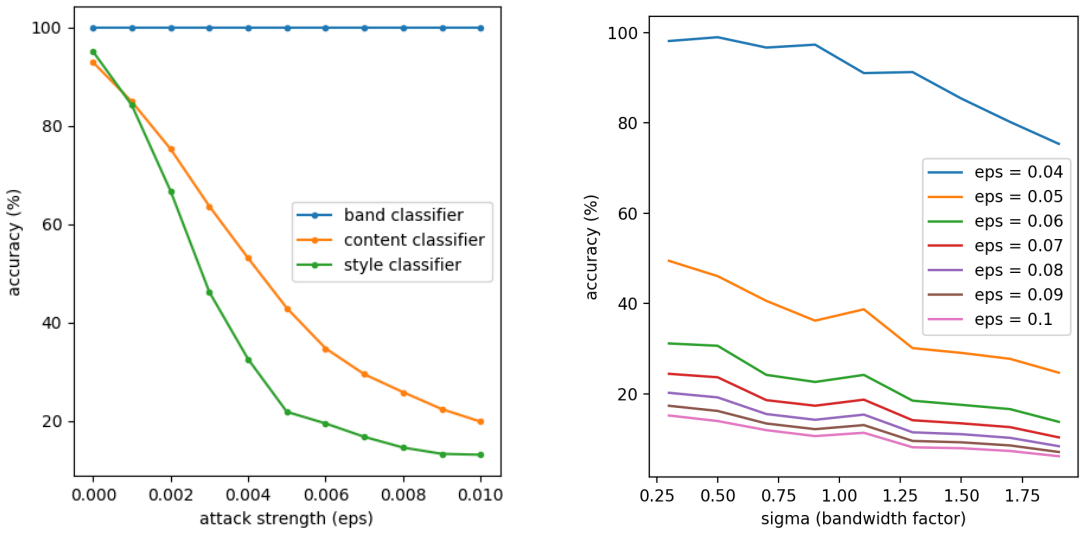
- *Content classes*: To match the settings from the style classes, we select 11 categories from the ImageNet dataset [66] and ensure that the initial accuracy on clean test samples are at a similar level as the one from the style classes. In general, a strict partial order may not exist between d_{intra} and d_{inter} for samples in the content classes, but the ratio of $d_{\text{inter}}/d_{\text{intra}}$ is tentatively in between those from the two extreme cases (i.e., style and band classes).

- *Band classes*: The band classes are built by equally dividing the input range $[0, 256)$ into 11 bands. For each band class, we generate random samples whose pixel values are independently sampled from the discrete uniform distribution with corresponding range. For instance, the pixel values in the first class are within the range of $[0, 24)$, the second class $[24, 48)$ etc. Under this extreme circumstance, we ensure that $d_{\text{intra}} < d_{\text{inter}}$ on average.

With the above unique settings and everything else controlled, the training process can be viewed as learning to partition the entire input space under the constraints from different sets of "anchor points" (i.e., training samples from distinctive distributions). The classifier needs to deform its decision boundary to meet the labels of the training samples, leading to a divergence on the geometry of the learned space.

Experiment: Each of the 11 categories in the three datasets has 1,000 images for training and 50 images for testing. All classifiers are trained from scratch based on the official ImageNet example code from PyTorch.[10] We choose ResNet-50 [111] with 11 neurons in the final FC layer as the architecture of the model. Except for a smaller batch size of 48, all other hyper-parameters remain the same as the default. The normalization procedure using mean and standard deviation calculated from ImageNet samples has been disabled

[10] `https://github.com/pytorch/examples/tree/master/imagenet`



(a) $\frac{d_{\text{inter}}}{d_{\text{intra}}}$: style < content < band

(b) increasing band overlap

Figure 4.11: Robustness of classifiers when categorizing same number of classes with different geometry.

for a consistent range of input domain and fair comparison.

After training, checkpoints that reach the highest test accuracy at the earliest epoch are passed to the second half of the experiment, in which we examine the adversarial robustness of the classifiers. We perform FGSM attack at various strengths to the test set and reevaluate the accuracy of those adversarial examples. Figure 4.11a shows the test accuracy on adversarial examples generated with varying strength for all three classifiers. The robustness of those classifiers agrees with the order of $d_{\text{inter}}/d_{\text{intra}}$ from the geometry of the data. Moreover, as the overlap among those bands increases, the ratio of $d_{\text{inter}}/d_{\text{intra}}$ tends to decrease; consequently, the robustness of the classifier will also decrease (Figure 4.11b). Classification on the non-overlapping band classes is considered "intrinsically" robust because of the linearly-separable distribution of categories.

Self-reflection: Although the band classes in Figure 4.10c may appear to be trivial, the style classes in Figure 4.10a are not. As most studies on adversarial examples are intoxicated by content classification (e.g., with ImageNet dataset), it is meaningful to consider an essentially distinctive classification problem in which the distribution of categories and geometry of decision boundaries are significantly different. Besides, to adequately justify

the influence of a specific factor, both positive and *negative* variations are required. The negative variation, however, is often missing in previous studies on adversarial defenses. Often when a defense is proposed, it is claimed to increase the robustness by mitigating a specific issue. The increase of accuracy on adversarial examples is sufficient to demonstrate the efficacy of a proposed defense. However, it may not be adequate to prove that the specified issue is indeed the determinant because we only observe the positive variation brought by the defense. To fully demonstrate the role of a claimed factor, the decrease of robustness is required when the issue is exacerbated. In the above verification process, we provide both positive (i.e., the band classes) and negative (i.e., the style classes) variations of the base scenario (i.e., content classes). Finally, a more rigorous way to create datasets for proving the hypothesis is to build a generator that can produce or evolve non-trivial data at a specified ratio of $d_{\text{inter}}/d_{\text{intra}}$, and we will leave it for future work.

4.6 Summary

In this chapter, we conduct several experiments that support our hypothetical image-space model and topological view of knowledge for single-label classification. Those experiments may appear to be separated and even unorthodox from a statistical perspective, but they all come across naturally following our understanding and philosophy. As a disclaimer, our understanding is not intended to be exclusive, and experiment results might also be well-interpreted from entirely different viewpoints. Moreover, we do not claim the state-of-the-art results as our settings are usually slightly different from classic benchmarks. The major contributions, therefore, are experiment designs and underlying rationale rather than the procedures and results. Nevertheless, we summarize the findings from our experiments.

- From the variational-calculus view of learning, data samples have a different and somewhat unpredictable impact on the training process. Therefore, dataset bias is an intrinsic property of machine learning, and it can happen even within the same

dataset.

- Training with more categories than necessary can boost the classification accuracy by utilizing information from "poorly justified true beliefs."

- The discrepancies between benign and adversarial examples are not irreconcilable. Adversarial examples can help preserve accuracy on benign samples via belief distillation. Even though beliefs on adversarial examples are false, they agree with the partition from true beliefs and are generalizable to benign samples.

- A monotonic relation exists between adversarial attacks and defenses. For a fixed attack, the successful defense rate first increases then decreases as the defense becomes stronger. For a fixed defense, the classification accuracy on adversarial examples first drops then rebounds as the number of attack iterations increases.

- The variance in adversarial robustness is large at both categorical and instance levels. It becomes possible to inflate accuracy (even to 100%) by selecting a large-scale validation set that allows a designated method to outperforms others.

- Under the guidance of the hypothetical image-space model, the following causes of adversarial examples are verified: path-connected regions, an excessive number of target categories, and the geometry of the categories.

The limitations of the experiment results are addressed as follows. The marginal increase in classification accuracy from an extra random-noise category only occurs to shallow networks at early epochs when no data augmentation is performed. For the experiments in Table 4.3, although the deep neural-network classifiers are trained with all fine-grained categories, the prediction is still limited to the target categories. Memory replay with belief distillation on adversarial noise fails in continual learning suggests that a gap still exists between natural images and random noise. In general, smoothing techniques are not considered widely effective for defending adversarial attacks as they only work

for high-frequency noise theoretically. We perform test-time smoothing in controlled experiments because their strength is straightforward and natural to measure. In addition, building sample-dependent geometric termination criteria for iterative approaches is difficult as the angle between two vectors in high dimensional space is very likely to be near $\frac{\pi}{2}$ [126]. Finally, illusive samples are only trackable with shallow networks. In cases of training deep neural networks with sufficient capacity for fitting all training data, the predictions for illusive samples often oscillate during early epochs.

CHAPTER 5
IMAGE-SPACE MODEL IN PHOTO STYLIZATION

In this chapter, we study the image-spaces in the context of photo editing and provide strategies that better guide the photo stylization process. Early research in this domain can be traced back to the studies on color spaces. In Section 5.1, we first verify the equivalence of various color spaces by examining the capability of neural networks in transforming images among various color spaces. Then in Section 5.2, we conduct empirical studies on photo style comprehension by revealing impacts from determinants that are beyond algorithms. Those determinants include choices of color spaces as well as descriptors and bases for measuring photo styles. Finally, we demonstrate two applications for photo-realistic style transfer in Section 5.3.

5.1 Color Space Transformation

Different color spaces model the image space differently by organizing color elements in distinctive manners. The organizations then lead to different distances between samples in the image space, which could result in divergent outcomes of the same algorithm. In this section, we examine the capability of neural networks with regards to transforming images among color spaces. Assuming that one color space is superior to others for a particular task, we would hope that neural networks can implicitly convert inputs to the optimal color space, thus support end-to-end training set-ups. We study the color space translation at two scales: pixel level and image level, using the workflow illustrated at Figure 5.1. . A neural network learns the mapping between two color spaces and translates inputs to the target space. The inverse transformation formula is then applied to convert the translated image back to the source space. Finally, reconstruction error is calculated by comparing the pixels in the original and reconstructed images in the RGB space.

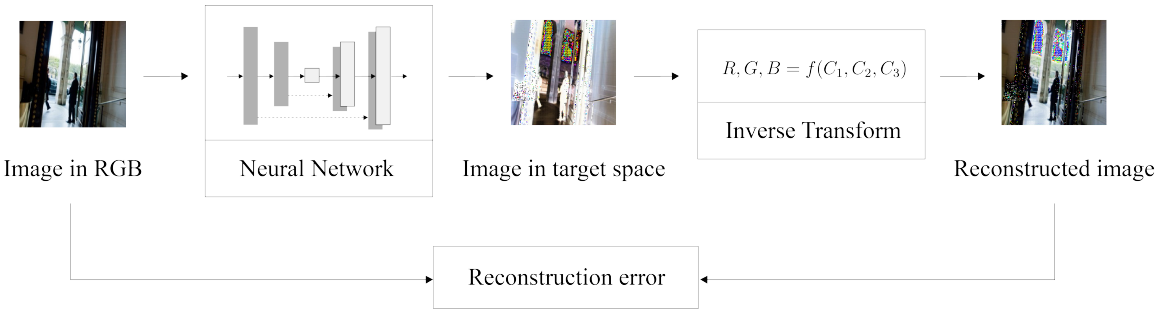


Figure 5.1: Workflow of the color space transformation experiments.

5.1.1 Pixel Level: Regression

We start with approximating the color-space transformation formulas at the pixel level and formulate the task as a regression problem. The goal of the neural network is to learn the transformation formula between two color spaces. A complete list of transformation formulas is available at [127], and the ones we adopt is provided at Appendix C. Each input-output pair consists of two 3-tuples from source and target spaces, respectively. A 3-tuple in a color space is referred to as a color element, and a single coordinate in the 3-tuple is called element component.

We conduct experiments to demonstrate the affinity between color spaces. Thus, instead of absolute values, the scale of errors is our focus. Among all $256^3 = 16,777,216$ possible elements in the RGB space, 10^6 unique samples are randomly selected as the sample set. Those samples are further divided into training and testing sets at the ratio of 8:2. For each experiment, we obtain an input-output pair by converting the sample elements to a source and target color spaces with corresponding closed-form formulas. A small neural network with 3 fully-connected layers (i.e., number of neurons: 64-32-3) and ReLU activation learns to convert a color element in the source space to the target space. We then apply the closed-form inverse transformation to the predicted value in the target space and calculate the reconstructed input in the source space. Finally, both the reconstructed and source inputs are converted back to the RGB space for error computation.

Table 5.1 shows the average root mean square error (RMSE) per element component

Table 5.1: Average root mean square error per element component between reconstructed and original inputs, measured in RGB color space with range 0 – 255. Larger values indicate larger gaps between the two color spaces. Row: source color spaces, column: target color spaces.

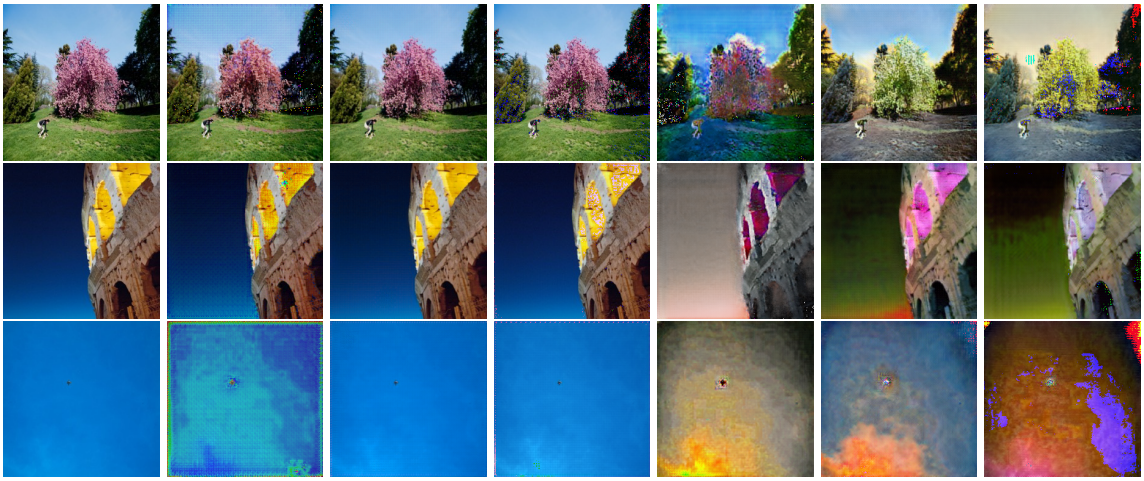
	RGB	HSV	HED	Lab	XYZ	YCbCr	YUV
RGB	–	8.35	0.65	0.81	1.20	0.06	0.13
HSV	2.03	–	3.29	2.68	5.47	2.06	2.20
HED	0.43	9.25	–	1.48	2.40	0.44	0.50
Lab	0.71	7.22	1.05	–	1.40	0.80	0.82
XYZ	1.23	11.71	1.05	1.74	–	1.01	0.96
YCbCr	0.08	12.78	0.53	1.04	1.16	–	0.34
YUV	0.26	8.40	0.46	0.84	0.91	0.31	–

(i.e., L_2 norm) when comparing the reconstructed and the original elements. For consistency, the error is measured in RGB color space with a range 0 – 255. Larger value in a cell indicates a larger gap between the two color spaces. During training, we noticed that the loss converges within 20 epochs with Adam optimizer [128] and default hyper-parameters. For most color spaces, learning the transformation formula is not hard with a small neural network: a 10 out of 255 difference is not obvious to our eyes, after all. Although the formulas may appear to be sophisticated in piecewise functions, the partial order of color elements is preserved in most cases. For HSV color space, however, color elements are reconstructed as a cylinder, leading to a different organization compared to the RGB cube. As a result, translating from and to HSV turns out to be more difficult.

5.1.2 Image Level: Translation

As pixel-level transformation seems easy for modern neural networks, we move forward to image-level transformation by formulating the task as an image-translation problem. Based on the official implementation[1] of pix2pix [77] and CycleGAN [78], we continue to validate whether the inverse transformation formula will remap the entire output image back to the original one. Unlike previous pixel-level experiments (in which each sample point is a 3-tuple color element), the sample points for the following image-level experiments are

[1] `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`



(a) Original (b) HSV paired (c) Lab paired (d) YCbCr paired (e) HSV unpaired (f) Lab unpaired (g) YCbCr unpaired

Figure 5.2: Qualitative results from image-level color space transformation. Except for the original images (first column) from MIT-Adobe FiveK dataset [87], all results are reconstructed by applying closed-form inverse transformation formula to outputs from the generator network whose goal is to learn the mapping from RGB to the target color space. Results with the label "paired" and "unpaired" are first transformed with pix2pix and CycleGAN, respectively.

images. For better visualization, we fix the source inputs in RGB color space and alter the target spaces for outputs.

During training, the generator network learns to map an image from RGB to the designated target space with the guidance signal generated from the discriminator network. One guidance signal from the discriminator is the self-supervision, which indicates the similarity between generated outputs and the real target outputs. In supervised mode (i.e., pix2pix), pixel-wised distance is available as extra guidance by comparing inputs with the paired outputs. In unsupervised mode (i.e., CycleGAN), a cycle-consistency loss is introduced to encourage the generated outputs to be remapped to the original inputs with less distortion, even though there is no paired output is available.

Figure 5.2 illustrates the original (5.2a) and reconstructed (5.2b – 5.2g) photos of the image-level translation. Results show that at the image-level, a generator network can almost fully capture the transformation between color spaces with pixel supervision, even

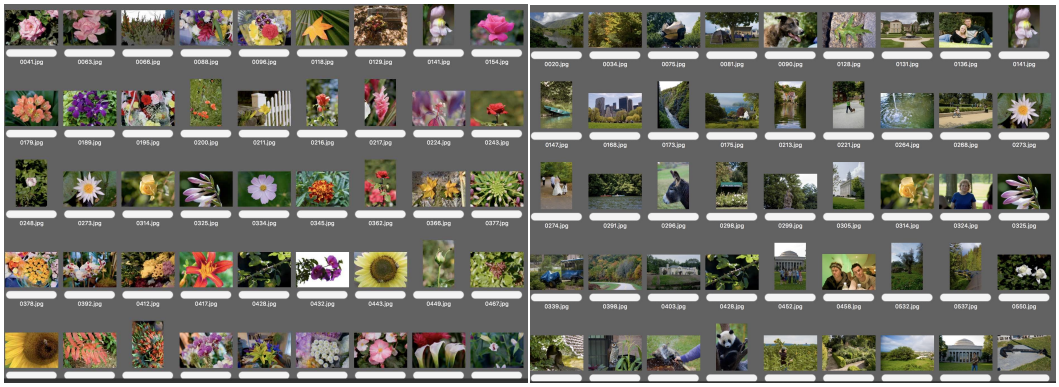
though noisy artifacts such as grids of blocks and impurities are observable (i.e., pixels in the sky). However, if no paired data is available, the generator fails to preserve color and photo-style information when it learns to transform images between color spaces. Such results imply that a good color model may not be required for supervised tasks, but it could be influential for unsupervised tasks related to photo styles.

5.2 Photo Style Comprehension

5.2.1 Hand-crafted Artistic Feature(s)

Different from painting styles which primarily focus on lower-level features such as texture (strokes/brushes) and absolute intensity, photo styles emphasize more on high-level abstract concepts such as harmony, distribution, and preference of color under an implicit assumption that the associated contents are controlled. In practice, pre-trained networks on ImageNet often serve as feature extractors for various tasks, including style transfer. Existing models in the model zoo, however, are optimized for tasks or sub-tasks of semantic segmentation and object detection, lacking the capability of describing color distribution with respect to photo styles. One naive approach to remove the structural information is pixel shuffling. Figure 5.3 shows sample clusters based on the features extracted via ImageNet pre-trained InceptionV3 [129]. With pixels shuffled, the feature extractor focuses more on color and generates clusters with consistent color tones rather than semantics. The features extracted from shuffled pixels, however, remain at the low level as they still rely on the color intensity.

To inject human-level abstraction for recognizing styles with AI, we introduce hand-crafted artistic features that better describe photos from an aesthetic viewpoint. Another concern is that when a target style is specified by a collection of photos rather than a single style image, those photos are geometrically scattered in the natural-image space as subjects vary. Consequently, it becomes difficult for a discriminator to connect them smoothly



(a) w/ original images (b) w/ pixel-shuffled images

Figure 5.3: Clustering on FiveK dataset [87] with InceptionV3 feature

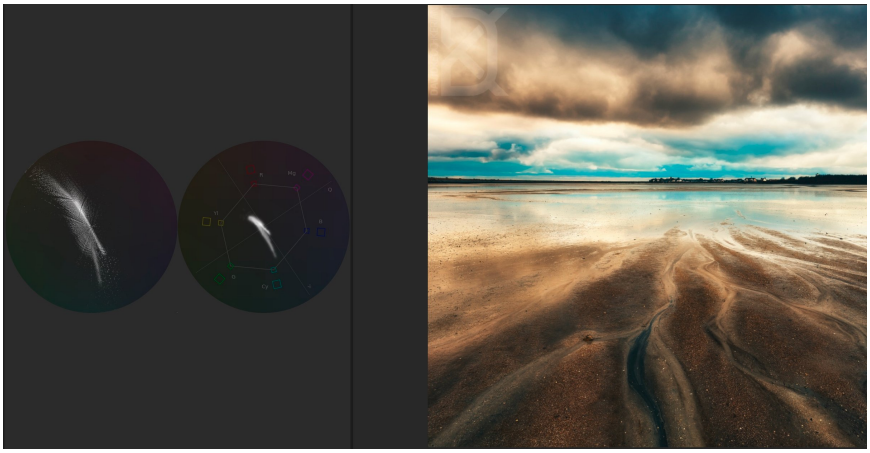
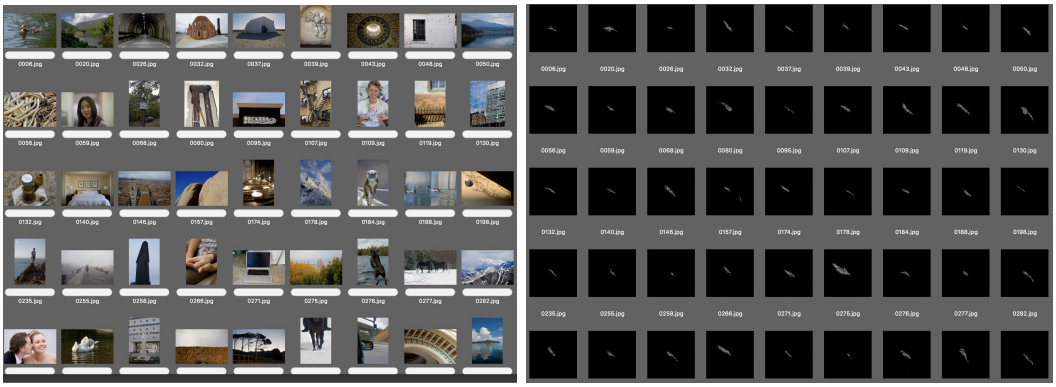


Figure 5.4: The diagonal pattern that is often observed in vectorscopes of artists' work. Left: vectorscopes in HLS and YUV, right: example photo.

with a continuous function. By adopting the subject-invariant vectorscope[2] feature, we can convert color information to shapes, which are easier for current pre-trained models to capture. The diagonal vectorscope pattern showed in Figure 5.4 often exists in artists' work. We then repeat the feature clustering experiment in Figure 5.3 by feeding the shapes in vectorscope to the pre-trained InceptionV3 and illustrate a sample cluster at Figure 5.5. Despite the variance in subjects, all photos in the sample cluster are composed of complementary colors and exhibit the diagonal pattern in their vectorscopes.

We further append the vectorscope feature as an additional plane of the inputs to the discriminator in the *exposure* framework [80]. The original feature planes consist of aver-

[2] `https://helpx.adobe.com/premiere-pro/using/using-waveform-monitors-vectorscope.html`



(a) Original images (b) Vectorscope in gray-scale mode

Figure 5.5: Image clustering with vectorscopes and InceptionV3 features

age luminance, contrast and saturation, all of which are filled with uniform values within each plane. The vectorscope plane, by contrast, equips with shape that are converted from the color distribution, which may simplify the judging process for the discriminator. Figure 5.6 illustrates the enhancement with the help of the additional vectorscope feature plane. Compared with the original baseline, the shape in vectorscope becomes wider and more stretched, leading to a more balanced color distribution with accurate white balance. It has to be pointed out that the enhancement is not guaranteed for all cases, and in a few cases, the final output becomes unnecessarily bright or saturated. Moreover, we may observe better results than the final output in the intermediate steps. Both the uncertainty in outputs from existing frameworks and the lack of style descriptors motivate us to build a metric that directly quantifies the styles of photos in the image space.

5.2.2 Remodeling Image Spaces with Artistic Presets

To better model the image space and assist AI in comprehending and quantifying photo styles, we build a preset classification benchmark dataset based on MIT-Adobe FiveK dataset [87] and the ImageNet [66] validation set. In particular, we apply ten artistic presets using Adobe Camera Raw to the five sets of expert-adjusted photos (i.e., artist A – E) in the FiveK dataset and all images from ImageNet validation set, which generate eleven styles (including the original) in each subset. Those styles then serve as the bases for measur-

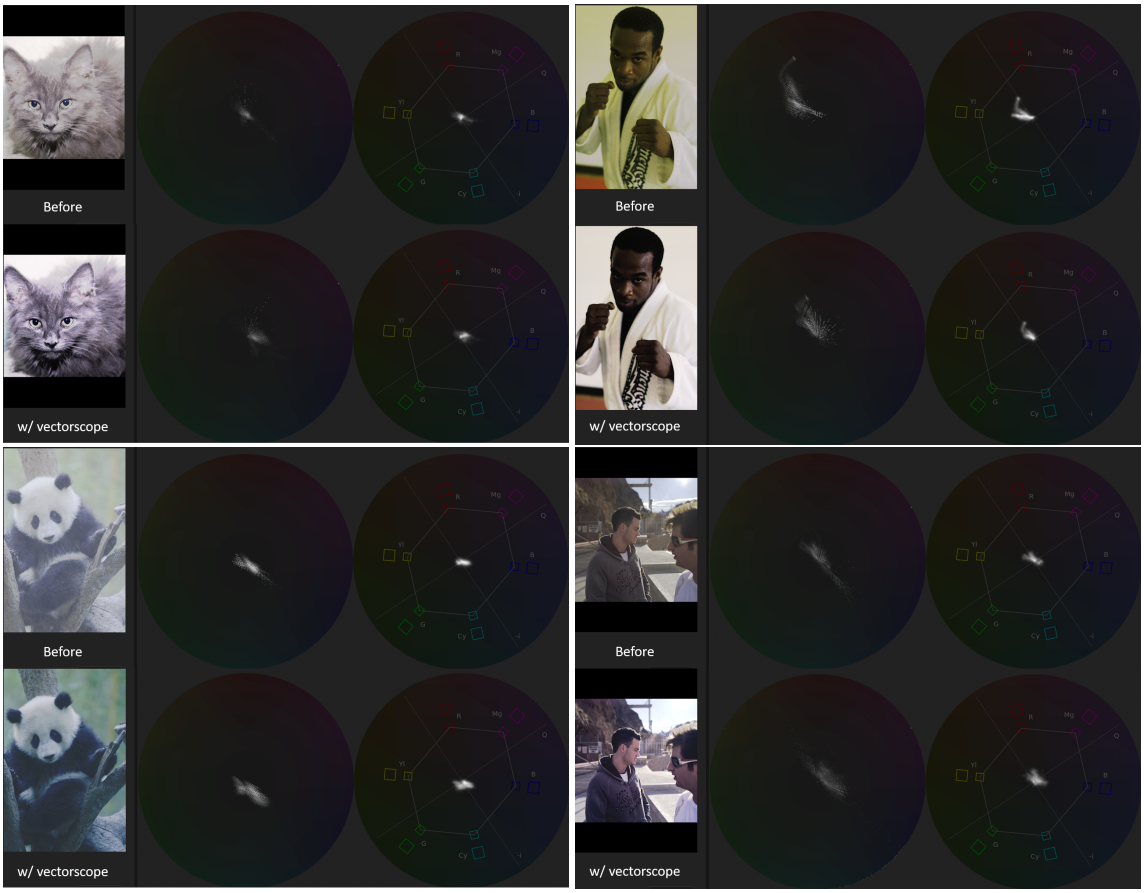
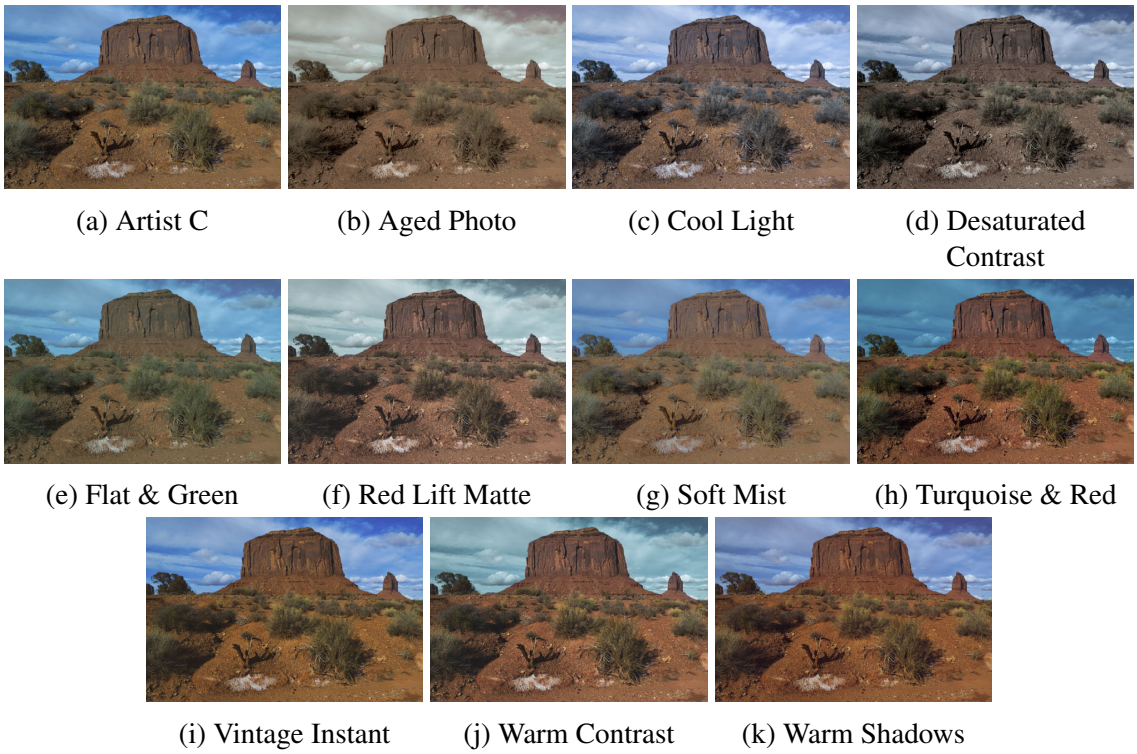


Figure 5.6: Enhancement in exposure framework with vectorscope as an additional feature plane for the discriminator. Within each example, the top row shows original results and the bottom row shows improved results with the help of vectorscope feature plane.

ing photographic styles. The rationale behind the set-up is that presets are non-destructive photo-editing techniques that disentangle styles from the preserved content. Moreover, presets are consistent photo styles and can serve as a baseline for AI comprehension. Figure 5.7 presents example photos and categories from the derived subset of FiveK-artist C.

One goal of this research is to disentangle and leverage both content and style recognition with AI, and it would be less desirable if part of the content is lost in stylized photos. To this end, before training preset classifiers, we first validate whether existing classifiers pre-trained on ImageNet are sufficiently robust to the non-destructive presets. Unfortunately, according to the results in Table 5.2, applying presets leads to accuracy drop for all pre-trained models and all presets. Models that reach higher accuaracies on the origi-



(a) Artist C (b) Aged Photo (c) Cool Light (d) Desaturated Contrast

(e) Flat & Green (f) Red Lift Matte (g) Soft Mist (h) Turquoise & Red

(i) Vintage Instant (j) Warm Contrast (k) Warm Shadows

Figure 5.7: Sample images from the preset dataset. The original photo in the example is from MIT-Adobe FiveK dataset [87], adjusted by Jaime Permuth (artist C). The derived categories are generated by applying artistic presets in Adobe Camera Raw.

nal validation set appear to be more robust to stylization. At the structure level, DenseNet outperforms others in terms of robustness and consistency, which is measured by the percentage of accuracy drop.

With the help of the preset dataset, we train a preset classifier on the derived subsets from FiveK - Artist C with complete supervision. Among all 5,000 content photos, we randomly select 4,000 of them and all their stylized versions as training data, and leave the rest as test data. As fine-tuning is out of the scope of this paper, we follow the default configurations and hyper-parameters from the ImageNet example provided by PyTorch[3], and choose ResNet50 [111] for all following experiments. Table 5.3 reports validation accuracy on all subgroups. For subgroups that are derived from other artists in FiveK dataset, the train-test split of the photos are matched to ensure that contents in test set are never observed during

[3] `https://github.com/pytorch/examples/blob/master/imagenet/main.py`

Table 5.2: Top-1 accuracy and accuracy drop on ImageNet validation set after applying presets, evaluated with ImageNet-pretrained models from torchvision.

Subset	VGG19 [130] w/ BN [131]	ResNet50 [111]	ResNet152 [111]
Original (baseline)	74.218, –	76.130, –	78.312, –
Desaturated Contrast	69.334, -6.581%	70.399, -7.528%	73.491, -6.156%
Cool Light	70.265, -5.326%	71.703, -5.815%	74.209, -5.239%
Turquiose & Red	72.735, -1.998%	74.573, -2.045%	77.018, -1.652%
Soft Mist	72.915, -1.756%	74.811, -1.733%	77.034, -1.632%
Vintage Instant	73.291, -1.249%	74.743, -1.822%	77.084, -1.568%
Warm Contrast	72.799, -1.912%	74.513, -2.124%	76.924, -1.772%
Flat & Green	72.455, -2.375%	73.967, -2.841%	76.518, -2.291%
Red Lift Matte	71.489, -3.677%	72.631, -4.596%	75.430, -3.680%
Warm Shadows	72.689, -2.060%	74.237, -2.487%	77.019, -1.651%
Aged Photos	68.828, -7.262%	70.271, -7.696%	73.651, -5.952%
Subset	DensNet161 [132]	Wide-ResNet101 [133]	MobileNet-v2 [134]
Original (baseline)	77.138, –	78.846, –	71.878, –
Desaturated Contrast	**75.626**, **-1.960%**	73.863, -6.320%	64.652, -10.054%
Cool Light	74.413, **-3.533%**	**74.525**, -5.480%	66.240, -7.845%
Turquiose & Red	76.002, **-1.473%**	**77.516**, -1.687%	69.757, -2.951%
Soft Mist	76.592, **-0.708%**	**77.736**, -1.408%	69.751, -2.960%
Vintage Instant	76.484, **-0.848%**	**77.680**, -1.479%	70.181, -2.361%
Warm Contrast	76.274, **-1.120%**	**77.376**, -1.864%	69.819, -2.865%
Flat & Green	76.484, **-0.848%**	**76.900**, -2.468%	69.326, -3.551%
Red Lift Matte	**75.736**, **-1.818%**	75.728, -3.955%	67.150, -6.579%
Warm Shadows	75.975, **-1.508%**	**77.175**, -2.119%	69.572, -3.209%
Aged Photos	**75.014**, **-2.754%**	73.669, -6.566%	64.080, -10.850%

training. As a continual discussion of the equivalance of color spaces in Section 5.1, the experiments are repeated in popular color spaces. Preset classification is not a hard task when training and testing data originate from the same data source. As for generalization, however, accuracy drops when original photos vary, especially when the source of data is changed (i.e., from FiveK to ImageNet). In that situation, preset classifiers trained in Lab color space possesses better ability to generalize. Moreover, all artistic color spaces we considered outperform the most commonly-used RGB space. The observation is further justified by another experiment at Table 5.4, in which we train classifiers to recognize the styles of five artists in the FiveK dataset and seven selected photo styles that focus on color distribution in the Flickr-80K [86] dataset.

[4]YCbCr does not work with black-and-white photos

Table 5.3: Preset classification accuracy on the derived preset dataset. Validation accuracies are tested on each subgroups (i.e., artist A – E, and ImageNet validation set). The models are trained on artist-C subset and in popular color spaces.

	5K-C	5K-A	5K-B	5K-D	5K-E	ImageNet
RGB	96.427	92.236	87.409	87.573	89.682	79.217
HSV	96.955	93.655	89.555	89.655	91.382	82.159
Lab	**97.509**	**94.182**	**91.073**	**90.709**	**92.155**	**83.489**
YCbCr	97.100	93.755	89.573	89.736	91.300	80.885

Table 5.4: Validation accuracy: classification of (1) Adobe-MIT FiveK artist styles and (2) seven selected Flickr-80K styles. The experiments are repeated in popular color spaces and identical train/test splits.

	RGB	HSV	Lab	YCbCr
FiveK	43.280	45.880	**46.720**	45.160
Flickr-80K	62.798	**63.426**	63.058	60.221[4]

The experiment results from Tables 5.3 and 5.4 imply interesting characteristics of photo styles. Geometrically, each photo represents a point in the image space, and each preset may correspond to a style vector that shifts a photo in a particular direction. When original photos (those serve as neutral negatives before applying presets) vary, the initial positions of points in the image spaces also differ. A preset vector that used to lead a photo from 5K-C to a particular category may transfer a photo from ImageNet validation set to another because the starting positions of the two photos differ, which explains larger errors from other subsets than 5K-C in Table 5.3. In the case of artist-style recognition, the style vectors (i.e., fields) are no longer consistent or position-invariant, resulting in more challenging styles and lower classification accuracy in Table 5.4. Finally, when training data are no longer paired, and each content only wears one style, recognizing photo styles can be considered as a partition of the entire input space. The seven photo styles (i.e., bright, ethereal, hazy, HDR, minimal, noir, and pastel) selected from Flickr-80K are distinct and comparably consistent with respect to color distribution. When all styles are involved, classification accuracy becomes less than 40%. It is, however, unclear whether styles should be interpreted as *positions* or *vector fields*, and we leave the problem for feature work.

5.2.3 Comparison on Feature Representations of Contents and Styles

As one objective of the previous experiments is to disentangle style from content, the features learned by preset classifiers are expected to be substantially different from those provided by ImageNet-pretrained models. Figures 4.10b and 4.10a illustrate the different partitions according to ImageNet-pretrained models and our preset classifiers. For semantic-related tasks such as ImageNet recognition challenge, a network learns to group samples according to content such as shape and texture. On the contrary, in our preset classification benchmark, the model is trained to cluster inputs according to styles such as color, although the samples might be farther away in Euclidean distance. Consequently, transfer learning with fixed ImageNet-features and a fully-connected layer (i.e., logistic regression) only reaches 61% accuracy for preset classification. Moreover, the accuracy cannot be notably enhanced by adding more non-linearity (e.g., 63% with a SE layer [135]) to the classifier because the required feature representations are significantly different for those two tasks.

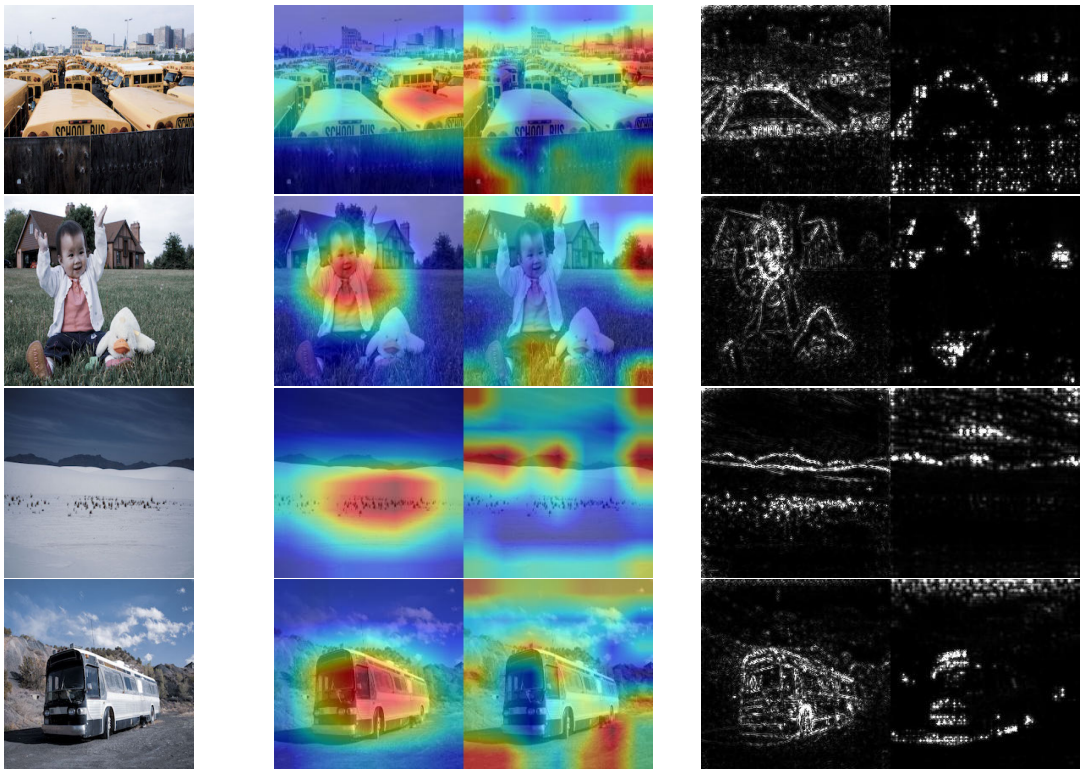
For better visualization, we apply PCA [136] and t-SNE [137] to features that are extracted from a test set with both ImageNet classfier and our preset classifier. Each color of the image frames in Figure 5.8 represents one of the eleven preset categories which is illustrated in Figure 5.7. Based on the features extracted with our preset classifier, photos with similar styles are clustered, even though their contents are notably different. In contrast, styles are almost indistinguishable under ImageNet-pretrained networks.

Figure 5.9 explains the prediction process of ImageNet and the proposed preset classifiers using Grad-CAM [138] and guided back-propagation [139]. The highlighted pixels indicate the most activated areas that trigger the prediction. The ImageNet classifier primarily focuses on instance(s) of objects (e.g., people, animals and vehicles), and the hot zones in the heat map are much more concentrated. In contrast, our preset classifier make predictions according to multiple areas that are dispersed in the image, which is more aligned with human judgement process as photo styles is more likely a global attribute than local. The predictions of ”Cool Light” category for example photos in Figure 5.9 are based on

(a) w/ ImageNet classifier

(b) w/ preset classifier

Figure 5.8: Visualization of feature clustering via t-SNE: each color in the image frame represents a preset category in the preset dataset.



(a) Inputs

(b) Grad-CAM
(left: ImageNet, right: preset)

(c) Guided BP
(left: ImageNet, right: preset)

Figure 5.9: Gradient visualization for ImageNet classifier and our preset classifier using Grad-CAM [138] [140] and Guided BP [139]. The target style is "Cool Light."

several cool-color regions, for example, sky, trees, blue jeans, and shadows.

The patterns in hot zones also imply the robustness of classifiers under input transforms. By default, training data are augmented with random resized-crop and horizontal flip. Table 5.5 shows validation accuracy under common input transforms. Even though the training data are not augmented with vertical flip, the preset classifier is still robust with respect to the transform because the color style remains consistent. When color information does change, such as in cases of color jitter and grayscale conversion, it is not surprising that our preset classifier stops working.

To further demonstrate the efficacy of the features from our preset classifiers, we adopt it as the descriptor in an image-retrieval task and compare the nearest neighbors with those that returned according to features extracted by an ImageNet-pretrained classifier. Example

Table 5.5: Robustness comparison between the ImageNet and our preset classifiers under common data transforms. Performance of the two classifiers is measured using validation accuracy and accuracy drop on ImageNet validation set and 5K-C preset test set, respectively.

	Horizontal flip	Vertical flip	Both flips
ImageNet	75.862, -0.352%	51.580, -31.895%	51.684, -31.759%
Preset	96.445, **0.019%**	96.045, **-0.396%**	96.082, **-0.358%**
	Color jitter	Grayscale	Baseline
ImageNet	60.114, **-20.686%**	64.718, **-14.638%**	76.130, –
Preset	18.364, -80.956%	23.782, -75.337%	96.427, –

results are presented at Figure 5.10. For each classifier, we first extract features for both the query and database using all but the last (i.e., classification) layers. Feature vectors of all images in the database are then sorted according to the Euclidean distance to the feature of the query. Finally, nine images with shortest feature distance to the query are returned. Because of the strong correlation between content and style in flickr-80k dataset, the results are often be similar. When results differ, however, our preset classifier better captures the distribution of colors whereas ImageNet classifier can be distracted by the content (e.g., insects, livestock, or large shapes). Moreover, results from our preset classifiers appear to be more visually consistent on photographic style (e.g., the black-and-white case).

5.3 Photo-realistic Style Transfer

5.3.1 Style-Aware Global Style Transfer

In this section, we demonstrate the benefit from the awareness of photographic styles provided by the preset classifier in the workflow of global style transfer. In an optimization-based style-transfer approach, the input content image is updated iteratively towards a reference style image until certain loss criteria are satisfied. For photo-realistic style transfer, an algorithm attempts to superimpose the color distribution of the target style image, meanwhile preserving the content and texture of the input. Therefore, to avoid potential destruction on content and texture during the pixel-wise update process, we customize the

(a) from the preset classifier (b) from the ImageNet classifier

Figure 5.10: Example results of image retrieval with descriptors provided by (a) our preset classifier and the (b) ImageNet-pretrained classifier. Query images are from ImageNet validation set and flickr-80k is used as database.

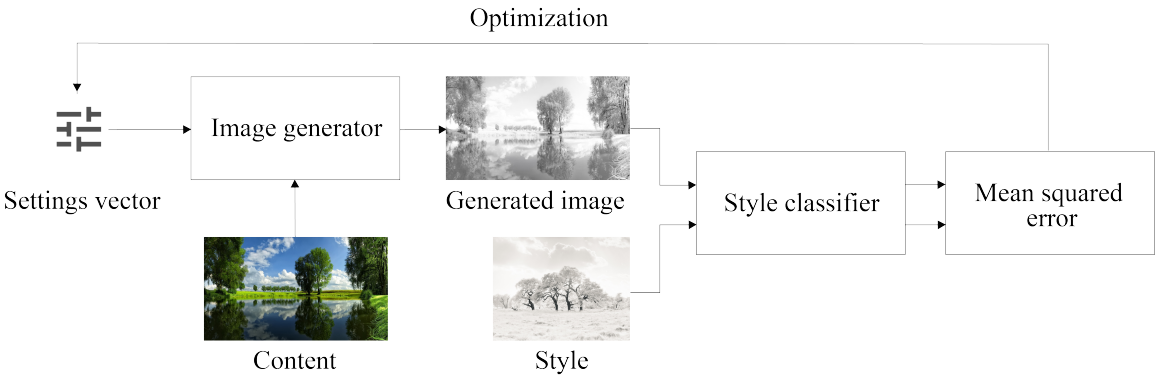


Figure 5.11: System architecture of the style-aware global style transfer

system as shown in Figure 5.11.

The image rendering process is achieved via a differentiable image pipeline similar to the one in [141], which takes both the content image and a settings vector as its input. The pipeline renders the output image in an end-to-end fashion by applying the global adjustments (e.g., brightness, contrast, hue) specified in the settings vector. Instead of directly optimizing the pixel values in the content image as often performed in artistic style transfer, we fix the content image and update the settings vector during each iteration. The method, therefore, is considered non-destructive and applies to images with various and large resolutions.

In artistic style transfer, loss criteria are built upon the VGG-19 style loss $\mathcal{L}_{\text{style}}$ [142]. In this work, we adopt the output logits from the top dense-layer of the preset classifier as a measurement of photographic style and compute the mean-square error between the logits from the generated photo and the target as a new loss $\mathcal{L}_{\text{preset}}$ in the criteria. The rationale behind the new loss term is that the categories in the preset classification benchmark can serve as bases for measuring unseen photographic styles, and useful guidance signals for the transfer process will be generated by utilizing the differentiation capability of the style-aware preset classifier. To demonstrate the benefits from $\mathcal{L}_{\text{preset}}$, we conduct controlled experiments and compare the results from different loss combinations.

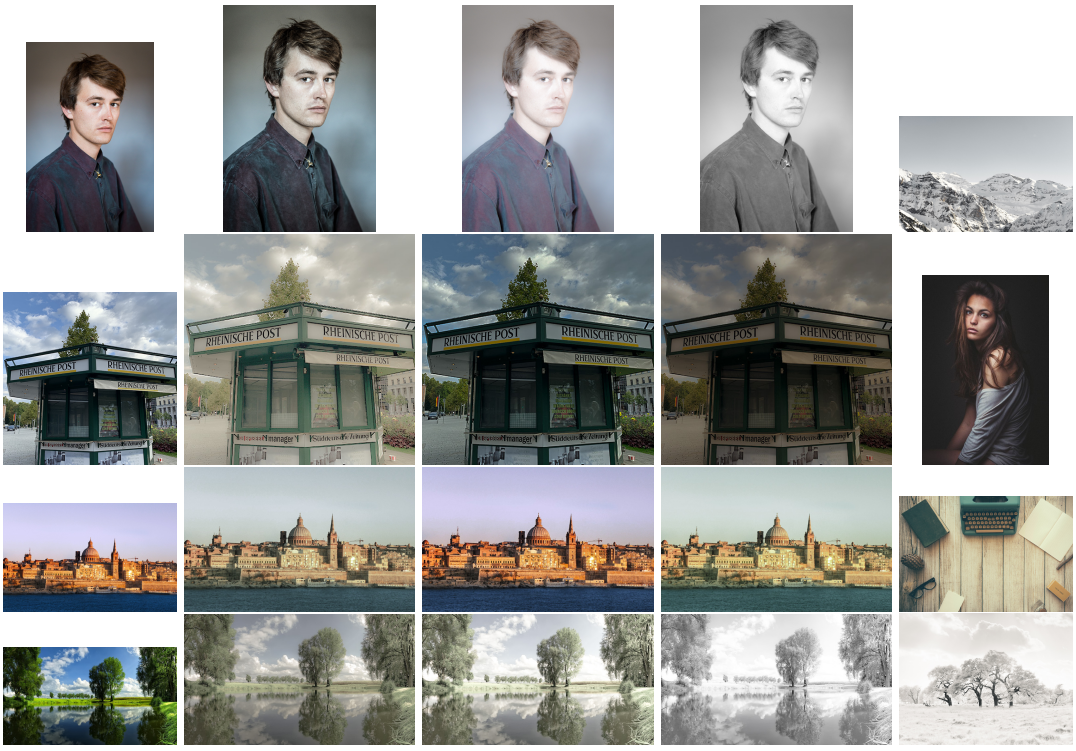
We begin with content-matched pairs of inputs and targets for numerical evaluation of

Table 5.6: Comparison of the performance between different loss combinations in global style transfer for content-matched inputs and targets.

Loss	Pixel MSE	Settings MSE
$\mathcal{L}_{\text{style}}$	0.0027	0.0902
$\mathcal{L}_{\text{style}} + \mathcal{L}_{\text{preset}}$	**0.0018**	0.067
$\mathcal{L}_{\text{preset}}$	0.0033	**0.034**

the results. For a given input under such an experiment setup, both the desired stylized version and the ground-truth settings vectors are available. During the experiments, the settings vectors are optimized through equal number of optimization steps. Table 5.6 reports the average mean squared errors between the final outputs and desired targets at both pixels and settings vectors. In specific, the settings MSEs are computed by comparing the optimized settings vectors and the ground-truth settings vectors. Similarly, the pixel MSEs are computed by comparing the target photos and the stylized photos which are generated by applying the optimized settings to the inputs. Table 5.6 shows that incorporating the preset loss $\mathcal{L}_{\text{preset}}$ results in more similar stylization compared with the ground-truth target photos and settings, possibly because the preset classifier is trained on images generated with the adjustments defined in the same settings space.

For more common use cases in practice, the contents in the inputs and targets are not matched. Therefore, the content loss $\mathcal{L}_{\text{content}}$ defined in [142] is introduced to preserve the content in the original inputs. It turns out that such a content loss compensates for substantial differences in exposure between the inputs and the targets by penalizing the settings that would remove the details in contents. A qualitative result comparison on various loss combinations is provided by Figure 5.12, in which the inputs and targets are not content-matched. With the same number of optimization steps, introducing the preset loss $\mathcal{L}_{\text{preset}}$ enables faster convergence towards the target style. Even though the preset loss $\mathcal{L}_{\text{preset}}$ by itself my not fully capture the hues (5.12b), combining the it with the style loss $\mathcal{L}_{\text{style}}$ leads to more comprehensive understanding towards the target styles and stylized photos which better resemble the targets (Figure 5.12c).



Content (a) $\mathcal{L}_{\text{style}}$ (b) $\mathcal{L}_{\text{preset}}$ (c) $\mathcal{L}_{\text{style}} + \mathcal{L}_{\text{preset}}$ Style

Figure 5.12: Results of global photo style transfer with various loss combinations for the cases when inputs and targets have differing contents. $\mathcal{L}_{\text{content}}$ is added in all combinations. Example photos are selected from the DPST dataset [143].

5.3.2 Content-Aware Local Style Transfer

Besides global processing for the entire photo, professional photographers and retouchers often refine their artwork using local adjustments. The local retouching reflects a paradigm shift from image-based to object-based photo editing. Instead of regarding each image as a unit sample in an image space or distribution, we consider each type of object (e.g., sky, sea, mountains) separately, assuming that each corresponds to a sub-space of its own. As a consensus has not been reached on photo-style measurement, we follow the exemplar-based configuration for local style transfer. The configuration naturally arises from an everyday use case where customers wish to produce their work similar to a masterpiece from experts by imitation. In this subsection, we present our combined content-aware style-transfer approach that supports local retouching on an object basis. As shown in Figure 5.13, the

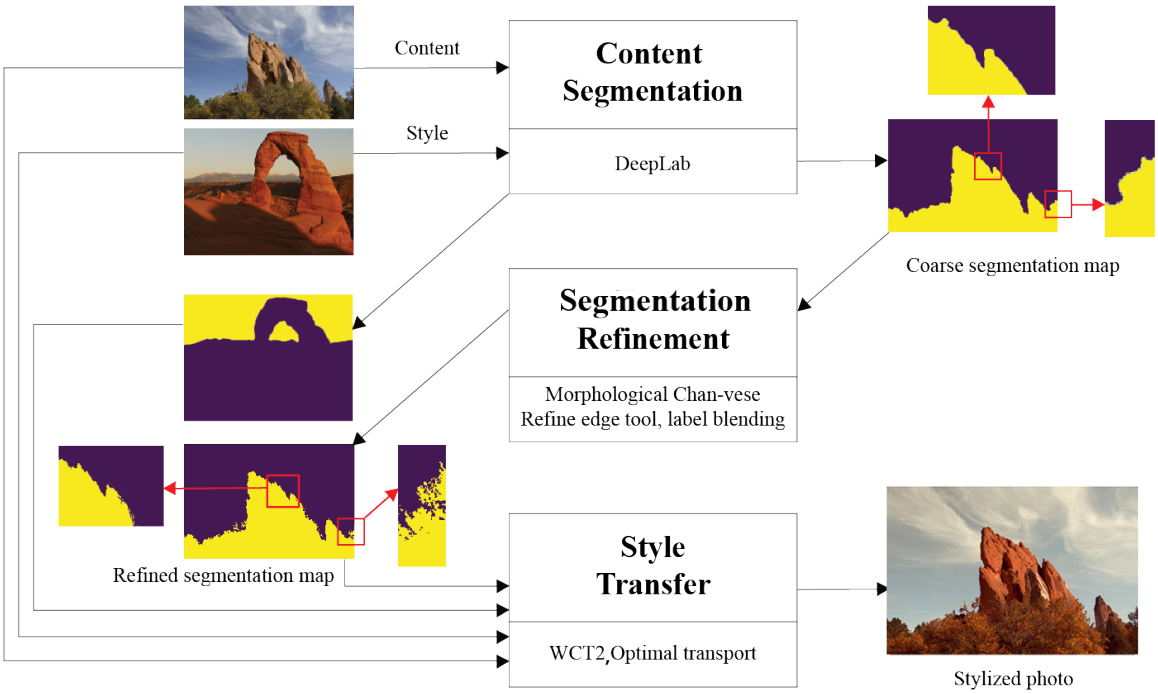


Figure 5.13: Workflow of the content-aware local style transfer.

proposed pipeline consists of three major components: *content segmentation*, *segmentation refinement*, and *style transfer*. In the following paragraphs, we will first introduce the selected methods for content segmentation and style transfer. Then in the edge refinement module, we focus on resolving the challenging issue from the desire for extreme segmentation accuracy. The proposed pipeline can be either applied as a complete method or a generator of reference image for subsequent non-destructive adjustments using the Adobe Lightroom ecosystem.

Content Segmentation

As subjects and compositions of a photo vary significantly according to the category of photography, it is unrealistic to achieve robust segmentation with a single model trained on a single dataset. In this work, we limit our scope of input photos to the landscape category and adopt the pre-trained DeepLabV3 [144] model for content segmentation. The segmentation model[5] was pre-trained on ADE20K dataset [145], which include 35 stuff classes

[5] `github.com/tensorflow/models/blob/master/research/deeplab/g3doc/model_zoo.md`

(e.g., sky, road, building) that are often observed in landscape images. The DeepLabV3 model provides coarse segmentation maps for both the input image and the exemplar image, which contains target styles. With indices of the semantic classes matched, style transfer is performed separately for each class.

Style Transfer

We employ a closed-form style transfer algorithm based on the Whitening and Coloring Transform (WCT) [146], which matches feature correlations of the input content image to those of the target style image via two projections. Following the notations from PhotoWCT [81], we let H_C and H_S demote the vectorized features of the input content image I_C and target style image I_S, respectively. The features for both content and style images are extracted by layers in a pre-trained auto-encoder for general image reconstruction purposes. The encoder $\mathcal{E}$ is fixed and borrowed from a VGG19 [130] model pre-trained on ImageNet. During training, only weights in the decoder $\mathcal{D}$ (which is symmetrical to the encoder and contains unpooling layers) will be learned.

Given zero-meaned content feature H_C and target style feature H_S, The stylized feature H_{CS} is computed via:

$$H_{CS} = P_S P_C H_C = \left(E_S \Lambda_S^{\frac{1}{2}} E_S^T\right)\left(E_C \Lambda_C^{-\frac{1}{2}} E_C^T\right) H_C \tag{5.1}$$

in which $P_C = E_C \Lambda_C^{-\frac{1}{2}} E_C^T$ and $P_S = E_S \Lambda_S^{\frac{1}{2}} E_S^T$ denote the whitening and coloring transform, respectively. The two diagonal matrices Λ_C and Λ_S are composed of eigenvalues of the covariance matrices $H_C H_C^T$ and $H_S H_S^T$, satisfying $H_C H_C^T = E_C \Lambda_C E_C^T$ and $H_S H_S^T = E_S \Lambda_S E_S^T$, respectively. Correspondingly, we have E_C and E_S which contain the orthonormal eigenvectors, satisfying $E_C E_C^T = E_S E_S^T = I$. The core idea of WCT is to establish the correlation between the stylized content feature and the target style feature by ensuring $H_{CS} H_{CS}^T =$

$H_S H_S^T$. The equality can be proved by the following derivations.

$$
\begin{aligned}
H_{CS} H_{CS}^T &= \left(E_S \Lambda_S^{\frac{1}{2}} E_S^T E_C \Lambda_C^{-\frac{1}{2}} E_C^T H_C\right)\left(E_S \Lambda_S^{\frac{1}{2}} E_S^T E_C \Lambda_C^{-\frac{1}{2}} E_C^T H_C\right)^T \\
&= E_S \Lambda_S^{\frac{1}{2}} E_S^T E_C \Lambda_C^{-\frac{1}{2}} E_C^T H_C H_C^T E_C \Lambda_C^{-\frac{1}{2}} E_C^T E_S \Lambda_S^{\frac{1}{2}} E_S^T \\
&= E_S \Lambda_S^{\frac{1}{2}} E_S^T E_C \Lambda_C^{-\frac{1}{2}} (E_C^T E_C) \Lambda_C (E_C^T E_C) \Lambda_C^{-\frac{1}{2}} E_C^T E_S \Lambda_S^{\frac{1}{2}} E_S^T \\
&= E_S \Lambda_S^{\frac{1}{2}} E_S^T E_C (\Lambda_C^{-\frac{1}{2}} I \Lambda_C I \Lambda_C^{-\frac{1}{2}}) E_C^T E_S \Lambda_S^{\frac{1}{2}} E_S^T \\
&= E_S \{\Lambda_S^{\frac{1}{2}} [E_S^T (E_C I E_C^T) E_S] \Lambda_S^{\frac{1}{2}}\} E_S^T \\
&= E_S \Lambda_S E_S^T = H_S H_S^T
\end{aligned} \tag{5.2}
$$

For implementation, we adopt the state-of-the-art wavelet-corrected version named WCT^2 [82], which progressively transfers styles for features within a single pass. It is a fast end-to-end model that can process inputs at relatively high resolution (e.g., 1024×1024 without post-processing procedures. In the model, max-pooling and unpooling layers are replaced by wavelet pooling and unpooling layers, which can fully recover the original signal using component-wise transposed-convolution and summation. To better preserve the structural information in the content image, high-frequency components are skipped from the encoder layer to the mirrored decoder layer, leaving only the low-frequency component for the next encoding layer.

It is also noted that the proposed pipeline is flexible for incorporating other methods that do not rely on pre-trained neural networks, such as those based on optimal transport [147]. However, one issue remains as the content-based local style transfer requires extremely high accuracy on the content segmentation to avoid color defects near the boundary, sometimes even in the interior region. We will address the issue in the upcoming segmentation refinement module.

Segmentation Refinement

The essence of content-aware local style transfer is to treat objects differently within their sub-spaces. As the two transforms for pixels in different sub-spaces can differ significantly, color inconsistency and defects often emerge for pixels that are mislabelled in the segmentation map. To mitigate the color inconsistency, authors of PhotoWCT [81] proposed an additional post-processing procedure that encourages that similar pixels in the input content image are mapped to similar outputs. The similarity of pixels is measured by an affinity matrix whose metric is based on means and variances of pixels in a local window. The post-processing smoothing procedure can be written as a closed-form matrix transform. One shortcoming of the post-smoothing method is that it may introduce unnecessary distortions that reduce the sharpness of the stylized photo. Furthermore, the memory requirement for the matrix computation explodes as image resolution increases.

In this work, we propose to directly handle the inaccuracy in coarse segmentation maps using a pre-processing refinement step before the style transfer. Once the segmentation maps become more precise, the issue of color defects will be settled from the root. In addition, the final outputs remain sharp because highly-confident areas will refrain from smoothing. The proposed segmentation refinement module consists of two key steps: *candidates selection* and *weight blending*. In general, certain areas with high confidence of belonging to a particular segmentation class will resume regular style transfer as usual. Meanwhile, candidate pixels that are controversial and might be misclassified in the coarse segmentation map will undergo the weight blending process, which will soften the color inconsistency and defects.

Candidates selection: In the coarse segmentation map, each pixel is assigned with a label that indicates the class of content to which it belongs. To evacuate the ambiguity in the segmentation map and obtain those controversial pixels that might belong to a different content class, we apply level-set evolution using the Chan-Vese model [148] for each label. The Chan-Vese active contour without edges is a classic segmentation method by itself

that does not require training. Unlike edge-based variational approaches, the method can segment target contents with disconnected components; thus, it can cope with interior parts such as sky pixels that are immersed in tree leaves and branches. In our case, we apply the method to compute an ambiguity map that encodes all potential segmentation classes for each pixel. For each content label, the coarse segmentation map provides an initial level set from the binary mask for that label. Based on the binary mask, the interior regions are initialized with +1 whereas the exterior regions are initialized with checkerboard. The level set evolution is performed for each content label separately following the gradient decent[6]:

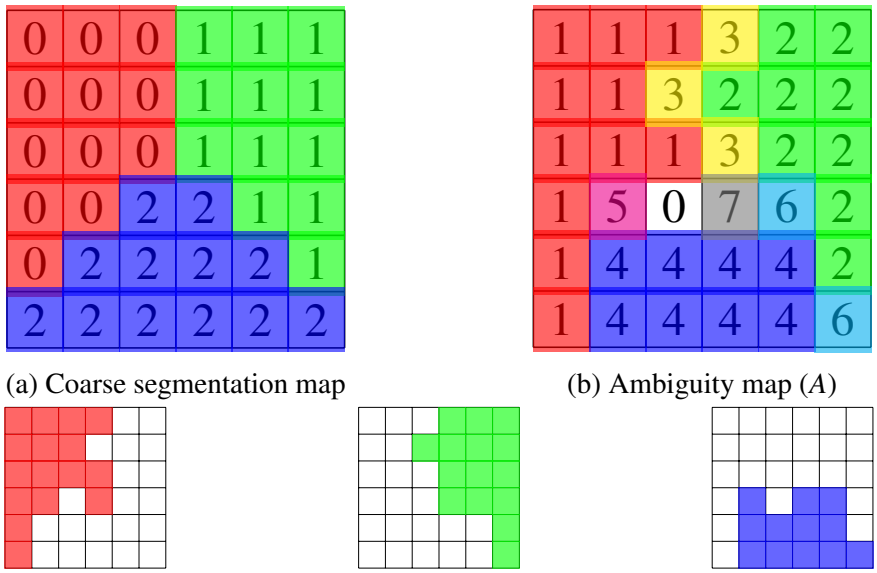
$$\frac{\mathrm{d}\varphi}{\mathrm{d}t} = \delta\left(\varphi(x)\right)\left[\mu\,\mathrm{div}\left(\frac{\nabla\varphi(x)}{|\nabla\varphi(x)|}\right) - \nu - \lambda_1(f-c_1)^2 + \lambda_2(f-c_2)^2\right] \tag{5.3}$$

in which f is a channel from the original image and δ the Dirac mass. μ, ν, λ_1, and λ_2 are real parameters. c_1 and c_2 are constants determined for segmentation. $\varphi(x) \in [-1, 1]$ is the level-set function, and the zero level-set $\varphi(x) = 0$ specifies the boundary between interior and exterior regions. As for implementation, we adopt the morphological version [149] of the Chan-Vese model provided by the scikit-image[7] package. After the level-set evolution, we obtain refined binary segmentation masks $\{\mathbb{1}_c\}$ for each content label c.

We now introduce a base-2 encoding scheme that combines those refined binary segmentation masks to an ambiguity map A. The ambiguity map specifies not only the controversial candidate pixels but also the relevant content classes to which the pixels might belong. The labels in the ambiguity map are computed by summing the refined segmentation masks with weight 2^c for each content label c in the segmentation map. Such an encoding process can be expressed as $A = \sum_c 2^c \cdot \mathbb{1}_c$. Figure 5.14 illustrates the process for computing the ambiguity map from refined segmentation maps. For clarification purposes, we highlight the difference between labels in a segmentation map and those in an ambiguity map. A label in the segmentation map indicates the content class to which the

[6] complete derivation provided at Appendix D

[7] `https://scikit-image.org/docs/dev/api/skimage.segmentation.html`



(a) Coarse segmentation map (b) Ambiguity map (A)

(c) Refined mask for 0 ($\mathbb{1}_0$) (d) Refined mask for 1 ($\mathbb{1}_1$) (e) Refined mask for 2 ($\mathbb{1}_2$)

Figure 5.14: Conversion from a coarse segmentation map to an ambiguity map which encodes all potential segmentation classes for each pixel with base-2 encoding.

pixel belongs, and the content label starts from 0. In contrast, a label in the ambiguity map is a code in base-2, which encodes the relevant content class(es) that a pixel might belong to, and the label starts from $1 = 2^0$. In addition, label 0 in an ambiguity map is reserved for pixels that are "abandoned" by all content classes after level set evolution, and a global style transform will be applied to those pixels. According to the ambiguity map, pixels with positive labels that are not a integer power of 2 are selected as candidates for weight blending. When converted to binary code, those labels from the candidates contain two or more 1's, and the positions of those 1's specify the labels of the relevant content classes. For example, the label $7(= 2^2 + 2^1 + 2^0)$ in the ambiguity map reflects that the pixels might belong to content classes 2, 1, and 0.

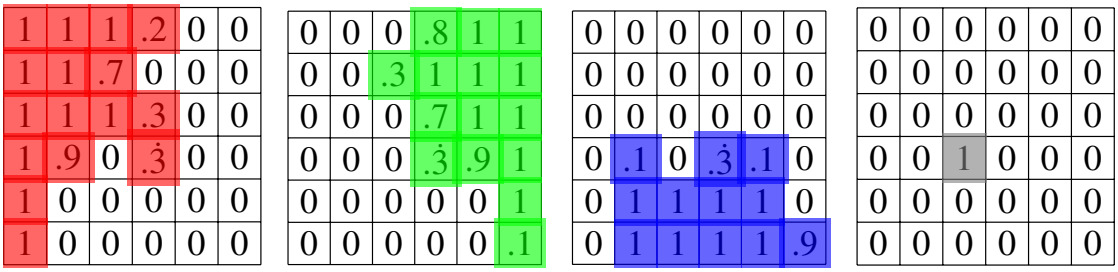
In practice, the level set evolution only applies to single-channel inputs by default. Thus, we evolve the level sets independently on four channels: red, green, blue, and grayscale. For each content label in the segmentation map, we compute two versions of the refined segmentation mask: (1) minimum mask $\mathbb{1}_{\text{min}}$ by taking the intersections of the results from all channels and (2) maximum mask $\mathbb{1}_{\text{max}}$ by taking the union. Alternatively,

Algorithm 2 Compute weight maps for each content class

Inputs:
A: ambiguity map obtained in the candidates selection step.
for all content class c in the segmentation map **do**
  Initialize the weight map W_c for content class c with value 0 and same size as A;
end for
for all label l in the ambiguity map **do**
  Compute index mask for label l: $\mathbb{1}_l = (A == l)$;
  Decode label l to binary code in which the position(s) of 1 specify relevant content class(es) $\{c\}$;
  if number of relevant class length($\{c\}$) == 1 **then**
    Set the weight to 1 at the masked indices: $W_c[\mathbb{1}_l] = 1.0$;
  else if number of relevant classes length($\{c\}$) == 2 **then**
    Divide the foreground c_f and background c_b classes according to the population of the content class: the background class contains more pixels;
    Apply matting to masked area $\mathbb{1}_l$ and obtain a probability map P w.r.t the foreground class;
    Assign weights for the foreground and background classes at the masked indices: $W_{c_f}[\mathbb{1}_l] = P$, $W_{c_b}[\mathbb{1}_l] = 1 - P$;
  else
    Equally distribute the weight for all relevant content classes if more than 2 classes are involved: $W_{c_i} = 1/n$, for $i = 1, \cdots, n$ when n classes are involved;
  end if
end for
Output: $\{W_c\}$: weight maps for each content label in the segmentation map.

the minimum mask and maximum mask can also be obtained by setting a threshold on the total counts of being True from all channels. For instance, $\mathbb{1}_{\text{min}}$ = True if more than half of the channels agrees and $\mathbb{1}_{\text{max}}$ = True when at least half of the channels vote for it. Intuitively, the minimum masks $\mathbb{1}_{\text{min}}$ specify regions that are confirmed to be correctly classified, whereas the maximum masks show all possible pixels for a particular content label. The ambiguity map is computed based on the maximum mask, and the minimum mask is devoted to protecting convinced pixels from unnecessary blending.

Weight blending: Given the ambiguity map, the weight blending step computes a weight map for each content label in the segmentation map. Values in a weight map of a particular content label will indicate the probability of pixels belonging to that content class. Algorithm 2 describes the process of computing weight maps for each content class



(a) Weight map for 0 (b) Weight map for 1 (c) Weight map for 2 (d) Global weight map

Figure 5.15: Weight maps for each content label in the segmentation map and the global

in the segmentation map. The algorithm is motivated by the workflow that professionals often follow when selecting subjects in Adobe Photoshop. From line 11 to line 13, the algorithm simulates the strokes with Refine-Edge Brush[8] and performs matting [150] to selected candidates. In our pipeline, we adopt the shipped version of Refine Edge Brush Tool, but other matting algorithms should also suffice. The matting algorithm takes as its input a trimap in which the foreground, background and unknown areas are specified. From those specification, the algorithm outputs an alpha matte which indicate the opacity (i.e., possibility) of the foreground. One crucial prerequisite for the matting algorithm is the availability of the trimap, which in turn guides the entire design of the segmentation refinement module. With the calculated ambiguity map, one can easily specify the three components required by the trimap. At line 12 of Algorithm 2, we define regions with ambiguity label 2^{c_f} as the foreground and regions with ambiguity label 2^{c_b} as the background. Naturally, candidate pixels with ambiguity label l form the unknown area for refinement. Figure 5.15 shows example weight maps that converted from the ambiguity map in Figure 5.14b. An additional global weight map is introduced to ensure that the sum of weights for each pixel equals 1.

With the weight maps obtained from the segmentation refinement module, the system performs style transfer with weight blending for photos and features at all network layers. Algorithm 3 documents the steps at each execution of the style transfer module. Depending

[8] `https://helpx.adobe.com/photoshop/using/select-mask.html`

Algorithm 3 Content-aware local style transfer with weight blending

Inputs:
H_C, H_S: input content and target style, they could be either images or features depending on the layer at which the transfer is performed.
M_S: coarse segmentation map for the target style photo.
$\{W_c\}, W_g$: weight maps for each content label and the global weight map.
Resize the weight maps $\{W_c\}, W_g$ to $\{\hat{W}_c\}, \hat{W}_g$ and segmentation mask M_S to $\hat{M}_S$ so that their sizes match those of current content H_C and style H_S, respectively;
for all class c in the content segmentation map $\{W_c\}$ **do**
    Compute index mask for input content feature w.r.t class c: $\mathbb{1}_C^c = (\hat{W}_c > 0)$;
    Compute index mask for target style image w.r.t class c: $\mathbb{1}_S^c = (\hat{M}_S == c)$
    Perform style transfer with the masked region: $H^c = \mathcal{F}\left(H_C[\mathbb{1}_C^c], H_S[\mathbb{1}_S^c]\right)$;
end for
Perform global style transfer: $H^g = \mathcal{F}(H_C, H_S)$;
Compute weighted average for the stylized output: $H_{CS} = \sum_c H^c \cdot \hat{W}_c + H^g \cdot \hat{W}_g$
Output: H_{CS}: stylized feature or photo, depending on the layer at which the transfer is performed.

on the layer at which the transfer is executed, the inputs could either be images or features extracted by a particular layer of the neural network. In other words, we perform feature blending in both image and feature domains.

Case Study and Result Comparison

At the end of this section, we provide case studies that illustrate the effectiveness of the proposed pipeline and compare results with existing commercial solutions. To begin with, Figure 5.16 and 5.17 show the benefits from content-awareness that we obtained from segmentation maps. Without clear correspondence for contents and objects, global style transfer might either be averaged out as results returned by the commercial software or overwhelmed by the overall color and tune with WCT^2. For the example in Figure 5.17, features are stylized at different layers in the auto-encoder. Results suggest that the style in the final output tends to be stronger once the style transfer takes place at the encoder layers. Such stronger styles are accompanied by a small amount of texture loss (e.g., pixels of the mountain and cloud).

If one watches closely, one may notice the color inconsistency on top of the mountain

Figure 5.16: Comparison between global style transfer with commercial software and content-aware local style transfer with our pipeline.

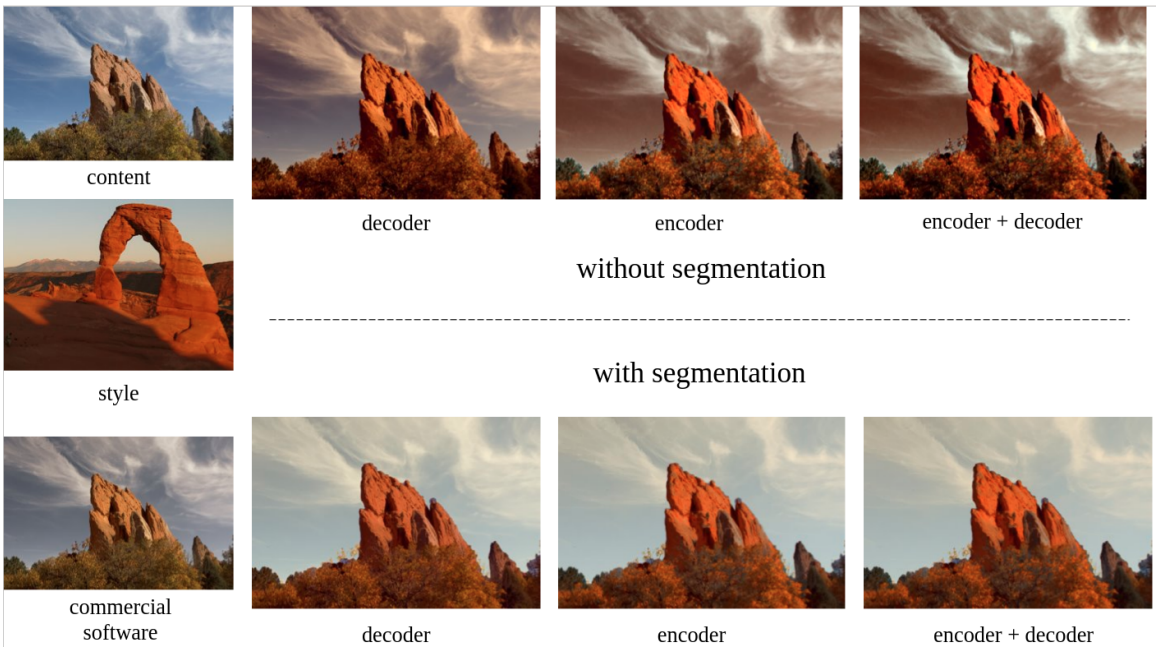


Figure 5.17: Example of content-aware local style transfer: features are stylized at different layers in the auto-encoder.

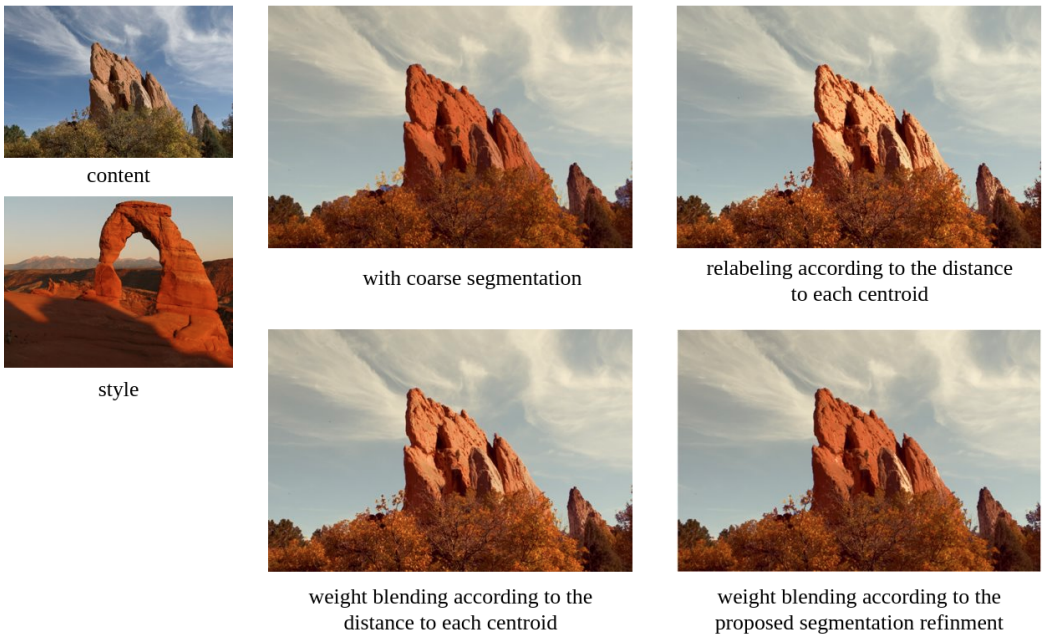


Figure 5.18: Comparison on segmentation refinement methods: naive relabelling and blending may lead to inaccuracy on pixels that used to be correctly classified whereas the proposed refinement approach corrects the defects and preserves correct classification.

(i.e., dark circle) and from sky pixels that are surrounded by tree branches (i.e., bottom-right corner). Those defects result from the inaccuracy of the coarse segmentation map. Naive solutions to tackle the issue include relabeling or blending the weights for each pixel according to the distance to each class centroid. Those naive solutions, although resolving defects for areas that used to be problematic, leads to potential inaccuracy on regions that were initially correct in the original segmentation map. Figure 5.18 compares the stylized photo when transfer with different segmentation refinement approaches. In cases of naive relabelling and blending, the mountain becomes brighter because their intensity values are closer to the averaged centroid of the sky category. On the contrary, the level set evolution in our segmentation refinement will keep most mountain pixels as a whole while resolving inconsistency at the mountain top and tree branches.

5.4 Summary

In this chapter, we study photographic style recognition and transfer under the guidance of the proposed hypothetical image-space model. As comprehending photographic styles is the first step towards reliable and explainable photo-realistic style transfer, we analyze critical factors for photographic style comprehension, including the choice of color spaces and bases for style measurement. Those factors are crucial at the design level and widely influential for subsequent tasks. In particular, differentiating and measuring photographic styles provides useful guidance signals for the global transfer process. By modeling the image space at the object level, we achieve content-aware photo-realistic style transfer, which supports local adjustments. The contributions of this chapter are summarized below.

- We verify the equivalence of various color spaces by examining the capability of neural networks with respect to translating images between color spaces. Results show that translation is easier with paired training data but hard with unpaired data.
- We revisit the photo-style recognition problem via image-space partition with presets and investigate the impact of color spaces in style recognition. Classifiers trained in the most-common RGB space are often outperformed by those trained in other color spaces such as HSV and Lab.
- We illustrate the differences between content and style classifiers via comparison in various tasks, including feature clustering, image retrieval, gradient visualization, and robustness under data transforms.
- We demonstrate that differentiating photographic styles provides useful guidance signals for global photo-realistic style transfer.
- We ensemble a content-aware local style transfer pipeline. The proposed segmentation refinement module removes defects from inaccurate segmentation maps and supports feature blending at various levels.

CHAPTER 6
DIRECTIONALLY PAIRED PRINCIPAL COMPONENT ANALYSIS FOR ESTIMATION OF COUPLED DATA

This chapter presents collaborative work on Directionally Paired Principal Component Analysis (DP-PCA) for the estimation of coupled data, which originates from inversion problems. Section 6.1 introduces the background and motivation of the research, highlighting the use case for the proposed method. Section 6.2 reviews traditional PCA methods for dimension reduction and estimation of coupled data, including standard independent PCA and joint PCA. In Section 6.3, we derive the proposed DP-PCA approach and explain the connection to other related algorithms. Finally, in Section 6.5, we demonstrate the effectiveness and superiority of the proposed DP-PCA approach in dimension reduction and estimation of coupled data via a series of experiments on data reconstruction and prediction.

6.1 Background and Motivation

Akin to the image-manifold assumption, the dimension of useful features for data samples in a dataset is generally much smaller than that of the data themselves, which alternatively implies that plenty of redundancy exists in data samples. Such redundancy and high dimension in data samples often lead to unnecessary complexity and issues such as the "curse of dimensionality" [1]. To mitigate those issues, various dimension reduction (or dimensionality reduction) approaches are invented to reduce the number of variables under consideration by obtaining a set of principal variables [151], among which Principal Component Analysis (PCA) [2] is the most widely used one.

Principal Component Analysis (PCA) is an unsupervised statistical approach that is primarily used for dimension reduction in various domains, including image processing,

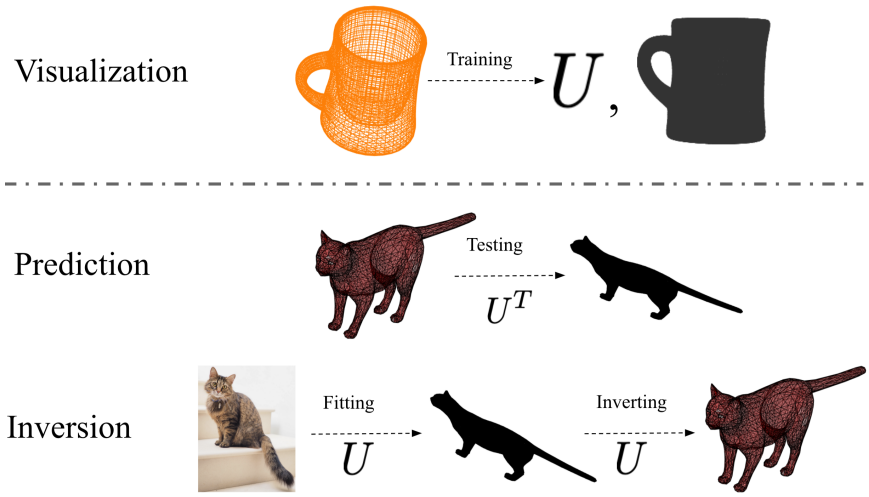


Figure 6.1: Common use cases of dimension reduction with PCA

data compression, data visualization, and pattern recognition. It summarizes the variation in potentially correlated multivariate attributes of features to a set of linearly uncorrelated components, each of which is a particular linear combination of the original variables. The extracted non-correlated components are called principal components (PC) and are estimated from the eigenvectors of the covariance matrix of the original variables. With the orthonormal basis defined by the set of principal components, PCA applies an orthogonal transformation to the data and projects them into a lower-dimensional subspace. The transformation can also be used for predictive modeling whereby the set of orthogonal basis forms the model and unseen data samples at test time can be approximated by a linear combination of those basis vectors.

6.1.1 Use Cases of Dimension Reduction with PCA

We may roughly categorize the usage of dimension reduction with PCA into three use cases: visualization, prediction, and inversion, and summarize the major difference of data flow in Table 6.1. The visualization process takes a single step that compresses the high-dimensional measurements to low-dimensional representations, which are possible for plots. In the prediction problem, a basis is learned during training, which supports transforming at test time measurements of data to a low-dimensional subspace. For data com-

pression and reconstruction, the reconstructed signal is computed via the inverse transform of the low-dimensional representation. When it comes to the inversion problem, however, the test phase *starts from the low-dimensional representation*, and an inverse transform is performed to map it to the corresponding high-dimensional measurements, which usually contain statistics of interest. In this research, we focus on the *inversion problem*, which differs from the previous two use cases in the following aspects.

One critical concept in the inversion problem is *raw data*, which are entities that can be quantified by various measurements, such as the aforementioned high-dimensional measurement/statistics X adopted in the PCA method. Although both the measurement X and representation A have fixed respective dimensions, the dimension of the raw data might not be fixed or even finite. The goal of the inversion problem is to compute the high-dimensional measurement X of the raw data D by leveraging the low-dimensional representation A, because computing X directly from D is extremely difficult and expensive. Unlike most datasets in prediction tasks, the dimension M of measurement X in the inversion problem is much larger than the number of data samples[1] N. Consequently, direct regression between the raw data and the high-dimensional measurements is infeasible and subject to overfitting. Alternatively, it is possible to compute[2] the low-dimensional representation A from the raw data D, thus obtaining the high-dimensional measurements X via the inverse PCA transform of A.

6.1.2 Motivations for the Proposed Directionally Paired PCA

In principle, PCA handles a single set of variables (i.e., measurements) by maximizing the variance with the principal components or equivalently minimizing the reconstruction errors. In scenarios of the prediction use case, we may have more than one set of correlated samples. For those situations, Canonical Correlation Analysis (CCA) [152], together

[1] e.g., in 3D reconstruction, we may have only 10 data cube of size 128^3 as training data.

[2] e.g., in 3D reconstruction, we can compute the shape (i.e., expansion coefficients) A by minimizing an energy function on D, which depends on the mean and principal components (weighted by A) learned from PCA.

with its general framework Partial Least Square (PLS) [153] methods, measures the linear correlation between two multi-dimensional variables by seeking a pair of bases such that the corresponding variables expressed in those bases are maximally correlated. Changing the objective to maximizing correlation, however, leads to sub-optimal bases in terms of maximizing variance and minimizing reconstruction errors for each respective set.

Our research on Directionally Paired PCA originates from a special scenario of an inversion problem in which we can access a pair of correlated datasets at training time, but can legitimately estimate the expansion coefficient for only one of them at test time. Henceforth, the two datasets are referred to as the observable and unobservable. The goal of the special use case is to conduct dimension reduction for both the observable and unobservable variables as well as providing high-quality reconstruction and predictions at test time. We further assume that reconstructing one would bring useful hints about the other because the two sets are correlated. In cases of both datasets being observable, we could develop two independent PCA models for each, by which we might ignore the correlation between the two sets and perhaps adopt higher dimensional representation than necessary. One naïve version that takes account the correlation between the two sets is called joint PCA, in which we stack both sets of samples and extract a single set of principal components. Such a technique forces the model to learn the correlations between the two while keeping the dimensionality lower than the sum of two independent PCA models. In joint PCA, the reconstruction for neither variable set is optimal because model's capacity is split between the two variable sets. Unfortunately, when one of the variable sets becomes unobservable, the budget and efforts spent on those variables will be in vain.

To overcome the above issue and achieve our goal for the special use case, we propose to combine the strength of PCA and correlation analysis within a single principled framework. We take advantage of PCA for the observable variables where it is possible to fit the low-dimensional coefficients at test time and utilize the correlation between the two for analyzing and representing the coupled unobservable part. The proposed Directionally Paired

PCA method can optimally express the variability of the observable part while maximally capturing the correlation between the two sets by sharing the expansion coefficients such that the fitting process of the observable part can be best applied to estimate the unobservable data. In the upcoming two sections, we present a formal description of the existing PCA techniques and describe our Directionally Paired PCA approach in detail.

6.2 Traditional Principal Component Analysis for Coupled Datasets

In this section, we establish notation and review how traditional PCA methods estimate coupled groups of variables in cases (1) where both are observable from a set of measurements (i.e., for independent PCA) and (2) where only one group is observable (i.e., for joint PCA). Let us assume that an $M_1 \times N$ matrix $X = [\mathbf{x}_1\, \mathbf{x}_2 \cdots \mathbf{x}_N]$ and an $M_2 \times N$ matrix $Y = [\mathbf{y}_1\, \mathbf{y}_2 \cdots \mathbf{y}_N]$ contain the separate collected components of pairs from N data measurements represented as vectors in $\mathbb{R}^{M_1}$ and $\mathbb{R}^{M_2}$. In particular, $\mathbf{x}_i$ and $\mathbf{y}_i$ represent the observable (high-confidence) and unobservable (low-confidence) components of the i^{th} data measurement, respectively. We further assume that the mean values of both sets of measurements are zero. If such a condition is not satisfied, the respective means should be pre-subtracted from each $\mathbf{x}_i$ and $\mathbf{y}_i$ for $i = 1, \cdots, N$.

6.2.1 Independent Principal Component Analysis

Standard PCA, applied independently to each groups of variables, yields independent L-dimensional subspaces of $\mathbb{R}^{M_1}$ and $\mathbb{R}^{M_2}$ that minimize the following mean squared error (MSE):

$$\varepsilon(A, U, B, V) = \frac{1}{N} \sum_{n=1}^{N} \left\| \mathbf{x}_n - \underbrace{\sum_{l=1}^{L} \mathbf{u}_l a_{ln}}_{U\mathbf{a}_n} \right\|^2 + \left\| \mathbf{y}_n - \underbrace{\sum_{l=1}^{L} \mathbf{v}_l b_{ln}}_{V\mathbf{b}_n} \right\|^2 \tag{6.1}$$

in which the columns of $M_1 \times L$ matrix $U = [\mathbf{u}_1 \cdots \mathbf{u}_L]$ and $M_2 \times L$ matrix $V = [\mathbf{v}_1 \cdots \mathbf{v}_L]$ denote orthonormal bases of the optimal L-dimensional subspaces, and the coefficients

$A = [\mathbf{a}_1 \cdots \mathbf{a}_N]$, $B = [\mathbf{b}_1 \cdots \mathbf{b}_N]$ with $\mathbf{a}_n = (a_{1n}, \ldots, a_{Ln})$, $\mathbf{b}_n = (b_{1n}, \ldots, b_{Ln})$ denote the L-dimensional vectors of coefficients for the linear combinations in these bases of the closest approximations to the measurements $\mathbf{x}_n$ and $\mathbf{y}_n$ in each collected pair. Noting that the orthogonal projections of $\mathbf{x}_n$ and $\mathbf{y}_n$ yield the best approximations for a given choice of subspaces U and V, we may eliminate the parameters a_{ln} and b_{ln} from the optimization problem by substituting $a_{ln} = \mathbf{u}_l^T \mathbf{x}_n$, $b_{ln} = \mathbf{v}_l^T \mathbf{y}_n$ (more compactly $\mathbf{a}_n = U^T \mathbf{x}_n$, $\mathbf{b}_n = V^T \mathbf{y}_n$ or even more compactly $A = U^T X$, $B = V^T Y$).

$$\varepsilon^*(U, V) = \frac{1}{N} \sum_{n=1}^{N} \left\| \mathbf{x}_n - \sum_{l=1}^{L} \mathbf{u}_l \mathbf{u}_l^T \mathbf{x}_n \right\|^2 + \left\| \mathbf{y}_n - \sum_{l=1}^{L} \mathbf{v}_l \mathbf{v}_l^T \mathbf{y}_n \right\|^2 \tag{6.2}$$

Expanding the squared norms into their constituent inner product terms (most of which vanish due to the orthogonality of $\mathbf{u}_1, \ldots, \mathbf{u}_L$ and $\mathbf{v}_1, \ldots, \mathbf{v}_L$) yields

$$\varepsilon^* = \frac{1}{N} \sum_{n=1}^{N} \left(\mathbf{x}_n^T \mathbf{x}_n - \sum_{l=1}^{L} \mathbf{x}_n^T \mathbf{u}_l \mathbf{u}_l^T \mathbf{x}_n \right) + \sum_{n=1}^{N} \left(\mathbf{y}_n^T \mathbf{y}_n - \sum_{l=1}^{L} \mathbf{y}_n^T \mathbf{v}_l \mathbf{v}_l^T \mathbf{y}_n \right) \tag{6.3}$$

Therefore, it is apparent that an equivalent optimization problem for $\mathbf{u}_1, \ldots, \mathbf{u}_L$ and $\mathbf{v}_1, \ldots, \mathbf{v}_L$ is to maximize

$$\sum_{n=1}^{N} \sum_{l=1}^{L} \mathbf{x}_n^T \mathbf{u}_l \mathbf{u}_l^T \mathbf{x}_n + \mathbf{y}_n^T \mathbf{v}_l \mathbf{v}_l^T \mathbf{y}_n \tag{6.4}$$

$$= \sum_{l=1}^{L} \mathbf{u}_l^T \left(\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T \right) \mathbf{u}_l + \mathbf{v}_l^T \left(\sum_{n=1}^{N} \mathbf{y}_n \mathbf{y}_n^T \right) \mathbf{v}_l \tag{6.5}$$

$$= \sum_{l=1}^{L} \mathbf{u}_l^T X X^T \mathbf{u}_l + \mathbf{v}_l^T Y Y^T \mathbf{v}_l \tag{6.6}$$

$$= \sum_{l=1}^{L} \|\mathbf{u}_l\|_{XX^T}^2 + \|\mathbf{v}_l\|_{YY^T}^2 \tag{6.7}$$

in which $\|\cdot\|_{XX^T}$ and $\|\cdot\|_{YY^T}$ denote the weighted L^2 norms via the positive definite[3] matrices XX^T and YY^T. As $\mathbf{u}_1, \ldots, \mathbf{u}_L$ and $\mathbf{v}_1, \ldots, \mathbf{v}_L$ must each be orthonormal, it is clear that

[3] positive semi-definite if X is not full rank.

the way to maximize this expression under this constraint is to choose the eigenvectors of XX^T and YY^T which, for each matrix, correspond to the L largest eigenvalues $\lambda_1, \ldots, \lambda_L$ (thereby yielding $\lambda_1 + \cdots + \lambda_L$ for each of the two pieces of the sum to be maximized). Those eigenvectors, as the choice of optimal basis vectors $\mathbf{u}_1, \ldots, \mathbf{u}_L$ and $\mathbf{v}_1, \ldots, \mathbf{v}_L$, are often called the first L *principal components* of each of the data sets X and Y, respectively.

In the independent PCA, the two orthonormal bases can fit data separately without capturing any correlations between the two datasets X and Y. However, if Y is completely unobservable such that we do not have access to measurements during fitting, we cannot use the second set of principal components. In case we had some way of capturing the correlations between the two datasets during training, we could have been able to use the reconstruction of X to gain some estimate of the unobservable Y. As it turns out, there is one modified approach that applies PCA to capture such correlations called joint PCA, which we describe in the following subsection.

6.2.2 Joint Principal Component Analysis

If we only allow a single set of linear combination coefficients $A = [\mathbf{a}_1 \cdots \mathbf{a}_N]$, so that the approximations of $\mathbf{x}_n$ and $\mathbf{y}_n$ in the respective bases U and V must always utilize the same set of expansion coefficients $\mathbf{a}_n$, then we may rewrite the energy by concatenating each $\mathbf{x}_n$ and $\mathbf{y}_n$ into a single measurement vector as well as concatenating each $\mathbf{u}_l$ and $\mathbf{v}_l$ into a single basis vector to yield an equivalent energy as follows.

$$\varepsilon(A, U, V) = \frac{1}{N} \sum_{n=1}^{N} \left\| \mathbf{x}_n - \sum_{l=1}^{L} \mathbf{u}_l a_{ln} \right\|^2 + \left\| \mathbf{y}_n - \sum_{l=1}^{L} \mathbf{v}_l a_{ln} \right\|^2 \tag{6.8}$$

$$= \frac{1}{N} \sum_{n=1}^{N} \left\| \begin{bmatrix} \mathbf{x}_n \\ \mathbf{y}_n \end{bmatrix} - \sum_{l=1}^{L} \begin{bmatrix} \mathbf{u}_l \\ \mathbf{v}_l \end{bmatrix} a_{ln} \right\|^2 \tag{6.9}$$

Assuming that these concatenated basis vectors $(\mathbf{u}_l, \mathbf{v}_l)$ are orthonormal, we may then write the orthogonal projections of the concatenated measurements $(\mathbf{x}_n, \mathbf{y}_n)$ as their closest

approximations to obtain

$$\varepsilon^*(U,V) = \frac{1}{N}\sum_{n=1}^{N}\left\|\begin{bmatrix}\mathbf{x}_n\\ \mathbf{y}_n\end{bmatrix} - \sum_{l=1}^{L}\begin{bmatrix}\mathbf{u}_l\\ \mathbf{v}_l\end{bmatrix}\begin{bmatrix}\mathbf{u}_l\\ \mathbf{v}_l\end{bmatrix}^T\begin{bmatrix}\mathbf{x}_n\\ \mathbf{y}_n\end{bmatrix}\right\|^2 \tag{6.10}$$

Similar to equation (6.3), expanding again the squared norms into their constituent inner product terms (most of which vanish due to the orthogonality) yields

$$\varepsilon^* = \frac{1}{N}\sum_{n=1}^{N}\left(\begin{bmatrix}\mathbf{x}_n\\ \mathbf{y}_n\end{bmatrix}^T\begin{bmatrix}\mathbf{x}_n\\ \mathbf{y}_n\end{bmatrix} - \sum_{l=1}^{L}\begin{bmatrix}\mathbf{x}_n\\ \mathbf{y}_n\end{bmatrix}^T\begin{bmatrix}\mathbf{u}_l\\ \mathbf{v}_l\end{bmatrix}\begin{bmatrix}\mathbf{u}_l\\ \mathbf{v}_l\end{bmatrix}^T\begin{bmatrix}\mathbf{x}_n\\ \mathbf{y}_n\end{bmatrix}\right) \tag{6.11}$$

It is apparent that an equivalent optimization problem for $\mathbf{u}_1,\ldots,\mathbf{u}_L$ and $\mathbf{v}_1,\ldots,\mathbf{v}_L$ is to maximize

$$\sum_{n=1}^{N}\sum_{l=1}^{L}\begin{bmatrix}\mathbf{x}_n\\ \mathbf{y}_n\end{bmatrix}^T\begin{bmatrix}\mathbf{u}_l\\ \mathbf{v}_l\end{bmatrix}\begin{bmatrix}\mathbf{u}_l\\ \mathbf{v}_l\end{bmatrix}^T\begin{bmatrix}\mathbf{x}_n\\ \mathbf{y}_n\end{bmatrix} = \sum_{l=1}^{L}\begin{bmatrix}\mathbf{u}_l\\ \mathbf{v}_l\end{bmatrix}^T\left(\sum_{n=1}^{N}\begin{bmatrix}\mathbf{x}_n\\ \mathbf{y}_n\end{bmatrix}\begin{bmatrix}\mathbf{x}_n\\ \mathbf{y}_n\end{bmatrix}^T\right)\begin{bmatrix}\mathbf{u}_l\\ \mathbf{v}_l\end{bmatrix} = \sum_{l=1}^{L}\begin{bmatrix}\mathbf{u}_l\\ \mathbf{v}_l\end{bmatrix}^T\begin{bmatrix}X\\ Y\end{bmatrix}\begin{bmatrix}X\\ Y\end{bmatrix}^T\begin{bmatrix}\mathbf{u}_l\\ \mathbf{v}_l\end{bmatrix} \tag{6.12}$$

The cost function is optimized when we choose $(\mathbf{u}_1,\mathbf{v}_1),\ldots,(\mathbf{u}_L,\mathbf{v}_L)$ to be the L largest eigenvectors of $\begin{bmatrix}X\\ Y\end{bmatrix}\begin{bmatrix}X\\ Y\end{bmatrix}^T$. Using such a technique, we force the correlations between the two datasets to be captured by the Joint PCA model. However, as we impose only a single set of linear coefficients, the model capacity is split between learning variations of both the datasets X and Y. When Y becomes completely unobservable at test time, joint PCA takes advantage of the correlation but loses the ability to optimally fit the observable data X. The loss of optimal fitting for the observable part X motivates us to design an asymmetric method that accounts for the correlation between X and Y, meanwhile optimally preserving information of the observable part X.

6.3 Proposed Asymmetric Principal Component Analysis

If we again allow only a single set of linear combination coefficients $A = [\mathbf{a}_1 \cdots \mathbf{a}_N]$ so that the approximations of $\mathbf{x}_n$ and $\mathbf{y}_n$ in the respective bases U and V must always utilize the same set of expansion coefficients $\mathbf{a}_n$, but impose the orthonormal basis U obtained using standard independent PCA on the data set X, together with the coefficients A that yield the best approximation of X within that bases (thereby minimizing the first term below over all choices of U and A as in standard PCA), then we may seek the "paired basis" V (not necessarily orthonormal) that minimizes the second term below given the choice of U and A.

$$\varepsilon_X(A, U) = \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{x}_n - U\mathbf{a}_n\|^2, \qquad \varepsilon_Y(A, V) = \frac{1}{N}\sum_{n=1}^{N} \left\|\mathbf{y}_n - V\mathbf{a}_n^*\right\|^2 \tag{6.13}$$

To obtain the optimal $\mathbf{a}_n^*(U)$, we differentiate ε_X with respect to $\mathbf{a}_n$ and setting the result to zero.

$$0 = U^T(\mathbf{x}_n - U\mathbf{a}_n^*) = U^T\mathbf{x}_n - \underbrace{U^TU}_{I}\mathbf{a}_n^* \tag{6.14}$$

$$\mathbf{a}_n^*(U) = U^T\mathbf{x}_n \tag{6.15}$$

Substituting the above $\mathbf{a}_n^*$ into ε_Y in equation (6.13) then leads to

$$\varepsilon_Y(V) = \frac{1}{N}\sum_{n=1}^{N} \left\|\mathbf{y}_n - V\mathbf{a}_n^*\right\|^2 \tag{6.16}$$

$$= \frac{1}{N}\sum_{n=1}^{N} \mathbf{y}_n^T\mathbf{y}_n - 2\mathbf{a}_n^{*T}V^T\mathbf{y}_n + \mathbf{a}_n^{*T}V^TV\mathbf{a}_n^* \tag{6.17}$$

$$= \frac{1}{N}\sum_{n=1}^{N} \mathbf{y}_n^T\mathbf{y}_n - 2\mathbf{x}_n^TUV^T\mathbf{y}_n + \mathbf{x}_n^TUV^TVU^T\mathbf{x}_n \tag{6.18}$$

Finally, differentiating $\varepsilon_Y(V)$ with respect to the matrix V, we obtain

$$\frac{\partial\varepsilon_Y}{\partial V} = -\frac{1}{N}\sum_{n=1}^{N} -2\mathbf{y}_n(\mathbf{a}_n^*)^T + 2V\mathbf{a}_n^*(\mathbf{a}_n^*)^T = -\frac{2}{N}\left(YA^T - VAA^T\right) \tag{6.19}$$

in which $A = \begin{bmatrix} \mathbf{a}_1^* & \cdots & \mathbf{a}_N^* \end{bmatrix} = \begin{bmatrix} U^T\mathbf{x}_1 & \cdots & U^T\mathbf{x}_N \end{bmatrix} = U^T X$. Setting the above derivative to zero yields the final expression of the paired bases:

$$V = YA^T(AA^T)^{-1} = YX^TU\left(U^TXX^TU\right)^{-1} \tag{6.20}$$

If we use the PCA basis U computed for X, then $U^TU = \mathcal{I}$, and $XX^TU = U\Lambda_X$, in which Λ_X represents the $L \times L$ diagonal matrix with the L largest eigenvalues of XX^T along the diagonal. Plugging those into equation (6.20) yields:

$$V = YX^TU\left(U^TXX^TU\right)^{-1} = YX^TU\left(\underbrace{U^TU}_{\mathcal{I}}\Lambda_X\right)^{-1} = YX^TU\Lambda_X^{-1} \tag{6.21}$$

Therefore during training, given a set of expansion coefficients $\mathbf{a}_n^*$ on optimal bases in U which estimate observable variables $\mathbf{x_n}$, the proposed method obtains an optimal prediction for unobservable variables $\mathbf{y_n}$ by applying the same weighted linear combination $\mathbf{a}_n^*$ to the matching basis elements in V. We refer to such a combination of traditional PCA for X and *unidirectional* correlation analysis for Y as Asymmetric Principal Component Analysis for paired datasets.

6.3.1 Relevant Linear Approaches for Estimation of Coupled Data

Canonical Correlation Analysis (CCA)

A well-known yet symmetric method that also produces a paired set of bases for a correlated pair of variables is Canonical Correlation Analysis (CCA). First introduced in [152], CCA manages to measure the linear relationship between two multi-dimensional variables $\mathbf{x} = (x_1, \cdots, x_{m_1})^T$ and $\mathbf{y} = (y_1, \cdots, y_{m_2})^T$. It seeks a pair of vectors $a \in \mathbb{R}^{m_1}$ and $b \in \mathbb{R}^{m_2}$ such that the linear combinations $u = a^T\mathbf{x} = \sum_{i=0}^{m_1} a_i x_i$ and $v = b^T\mathbf{y} = \sum_{j=0}^{m_2} b_j y_j$ maximize the correlation $\rho = \text{corr}\left(a^T\mathbf{x}, b^T\mathbf{y}\right)$. The random variables u and v in $\mathbb{R}$ are called the first pair of canonical variates. The methods then iteratively seeks pairs of canonical variates

that maximize the above correlation, subjecting to the constraint(s) that the new canonical variates shall be uncorrelated with previous canonical variates. The entire process can take up to $\min(m_1, m_2)$ iterations, and correlations between the canonical variates u and v indicate correlations among the terms $a_i x_i$ and $b_j y_j$.

Although both CCA and the proposed Directionally Paired PCA can project coupled data to a lower-dimensional subspace, the generated pair of bases are different because of different objectives. In CCA, correlation is maximized symmetrically by optimizing over all choices of bases for *both* X and Y, resulting in orthonormal bases in both U and V. In the proposed Directionally Paired PCA, however, the primary goal remains to be minimizing the reconstruction error by utilizing the correlation as much as possible. In particular, correlation is maximized only over the choice of V by sharing equally weighted expansions A associated with the basis U, whereas U itself is optimized independently of V with the different goal of maximizing its own variance (or equivalently minimizing the reconstruction error) across the training set X. A consequence of this asymmetric optimization is that while the PCA calculated basis U will be orthogonal, its paired correlated basis V will typically not be. As for calculation, both PCA and CCA can be formulated as solving eigenvalue equations with slightly different matrix coefficients [154, 155]. For CCA, the eigenvalue equation is:

$$\left(\begin{bmatrix} S_{xx} & 0 \\ 0 & S_{yy} \end{bmatrix}\right)^{-1} \begin{bmatrix} 0 & S_{xy} \\ S_{yx} & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \rho \begin{bmatrix} u \\ v \end{bmatrix} \tag{6.22}$$

in which S denotes the covariance matrix of the variables specified by its subscripts. In practice, CCA is applied primarily for modeling and correlation analysis, which tends to overfit data for reconstruction and prediction. Therefore, we adopt a customized version named Canonical Regression (CR) [156, 157], which performs additional regression analysis between **A** and **B** in the subspaces after dimension reduction (Figure 6.2b).

Partial Least Square Regression (PLSR)

The CCA approach can be considered the special mode "B" of a more general framework named Partial Least Square (PLS) methods [153]. The equivalence between CCA and orthonormalized PLS is further addressed in [158]. One slight difference between CCA and the Partial Least Square Regression (PLSR, mode "A") is that instead of maximizing the correlation between $\mathbf{X}$ and $\mathbf{Y}$ as in CCA, the objective function for maximization in PLSR becomes the *covariance* between X and Y [155]. Correspondingly, the eigenvalue equation becomes

$$\left(\begin{bmatrix} S_{xx} & 0 \\ 0 & S_{yy} \end{bmatrix}\right)^{-1} \begin{bmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \rho \begin{bmatrix} u \\ v \end{bmatrix} \tag{6.23}$$

The PLSR approach has an "opposite" motivation compared with the proposed Directionally Paired PCA. In Directionally Paired PCA, we assume that the predictor X remains observable with high confidence at all times and hope to predict the values of unobservable low-confidence variables Y at test time with the help of the correlation under the premise that X is optimally preserved. On the contrary, the goal of PLSR is to predict as accurately as possible the values of the important predictands Y (which might be expensive to capture) based on the less important predictors X (which are cheaper to capture) by utilizing the covariance between the two. It is, therefore, reasonable that the predictors X may not be optimally reconstructed when necessary.

Mathematically, the process can be expressed as follows by adapting[4] the notation from [159, 160] with matrix transpose and substitutions of symbols for consistency purpose.

$$\mathbf{X} = \mathbf{UA} + \mathbf{E}, \qquad \mathbf{Y} = \mathbf{VB} + \mathbf{F} \tag{6.24}$$

[4]In the original notation system of PLSR, data matrices are composed of row vectors, and transforms are expressed with right-multiplication. For consistency, we switch to column vectors and left-multiplications. To match the notation of PCAs in previous sections, we make the following substitutions to the original notation in PLSR: $\mathbf{P} \to \mathbf{U}$, $\mathbf{Q} \to \mathbf{V}$, $\mathbf{T} \to \mathbf{A}$, $\mathbf{U} \to \mathbf{B}$, $\mathbf{B} \to \boldsymbol{\beta}$.

in which $\mathbf{A}$ and $\mathbf{B}$ (both with size $L \times N$) are scores after dimension reduction (i.e., expansion coefficients as previously in PCAs) for original data $\mathbf{X}$ (with size $M_1 \times N$) and $\mathbf{Y}$ (with size $M_2 \times N$) after dimension reduction, respectively. $\mathbf{U}$ (with size $M_1 \times L$) and $\mathbf{V}$ (with size $M_2 \times L$) are the corresponding loadings for $\mathbf{X}$ and $\mathbf{Y}$, respectively. $\mathbf{E}$ (with size $N \times M_1$) and $\mathbf{F}$ (with size $N \times M_2$) are reconstruction errors between the approximation and the original data. Unlike PCA, in which both transforms and inverse transforms can be performed with a single pair of orthonormal bases U and V, PLSR requires an additional pair of weights matrices to transform input data to the lower-dimensional subspace. In contrast to the transforms $A = U^T X, B = V^T Y$ in PCA, we have

$$\mathbf{A} = \underbrace{\left(\mathbf{W}^T\mathbf{U}\right)^{-1}\mathbf{W}^T}_{\mathbf{X}^T_{\text{rotations}}}\mathbf{X}, \qquad \mathbf{B} = \underbrace{\left(\mathbf{C}^T\mathbf{V}\right)^{-1}\mathbf{C}^T}_{\mathbf{Y}^T_{\text{rotations}}}\mathbf{Y} \tag{6.25}$$

in which $\mathbf{W}$ (with size $M_1 \times L$) and $\mathbf{C}$ (with size $M_2 \times L$) are weights matrices for $\mathbf{X}$ and $\mathbf{Y}$, respectively. The two combined matrices $\mathbf{X}_{\text{rotations}}, \mathbf{Y}_{\text{rotations}}$ are called rotations that support the transformation from the input space to the L-dimensional subspace. In the low-dimensional subspace, a regression analysis is conducted between $\mathbf{A}$ and $\mathbf{B}$:

$$\mathbf{B} = \mathbf{R}\mathbf{A} \tag{6.26}$$

in which the obtained relation matrix $\mathbf{R}$ (with size $L \times L$) indicates the relation between the scores (i.e., $\mathbf{A}, \mathbf{B}$) of $\mathbf{X}$ and $\mathbf{Y}$.

During training, all of the above matrices (i.e., $\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{C}, \mathbf{A}, \mathbf{B}, \mathbf{R}$) can be calculated from $\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}$ iteratively with algorithms such as NIPALS [153]. At test time, we no longer have access to $\mathbf{Y}_{\text{test}}$ and the goal is to predict $\mathbf{Y}_{\text{test}}$ from $\mathbf{X}_{\text{test}}$. Starting with $\mathbf{A}_{\text{test}} = \mathbf{X}^T_{\text{rotations}}\mathbf{X}_{\text{test}}$, the estimation of the scores of $\mathbf{Y}$ (i.e., $\mathbf{B}$) is computed using $\hat{\mathbf{B}}_{\text{test}} = \mathbf{R}\mathbf{A}_{\text{test}}$. We can eventually predict the values of the unobservable $\mathbf{Y}_{\text{test}}$ by mapping its estimated scores $\mathbf{B}_{\text{test}}$ to the original high-dimensional data space with the corresponding loadings matrix

$\mathbf{V}$ as $\hat{\mathbf{Y}}_{\text{test}} = \mathbf{V}\mathbf{B}_{\text{test}}$. To be more concise for predicting $\mathbf{Y}$ with $\mathbf{X}$, we may combine those involved matrices to a single regression coefficient $\boldsymbol{\beta}$ (with size $M_2 \times M_1$) as:

$$\boldsymbol{\beta} = \underbrace{\mathbf{V}\mathbf{R}}_{V \text{ in DP-PCA}} \underbrace{\left(\mathbf{W}^T\mathbf{U}\right)^{-1}\mathbf{W}^T}_{\mathbf{X}^T_{\text{rotations}}} \tag{6.27}$$

We can then predict the values of unobservable $\mathbf{Y}_{\text{test}}$ with the regression coefficient $\boldsymbol{\beta}$ as:

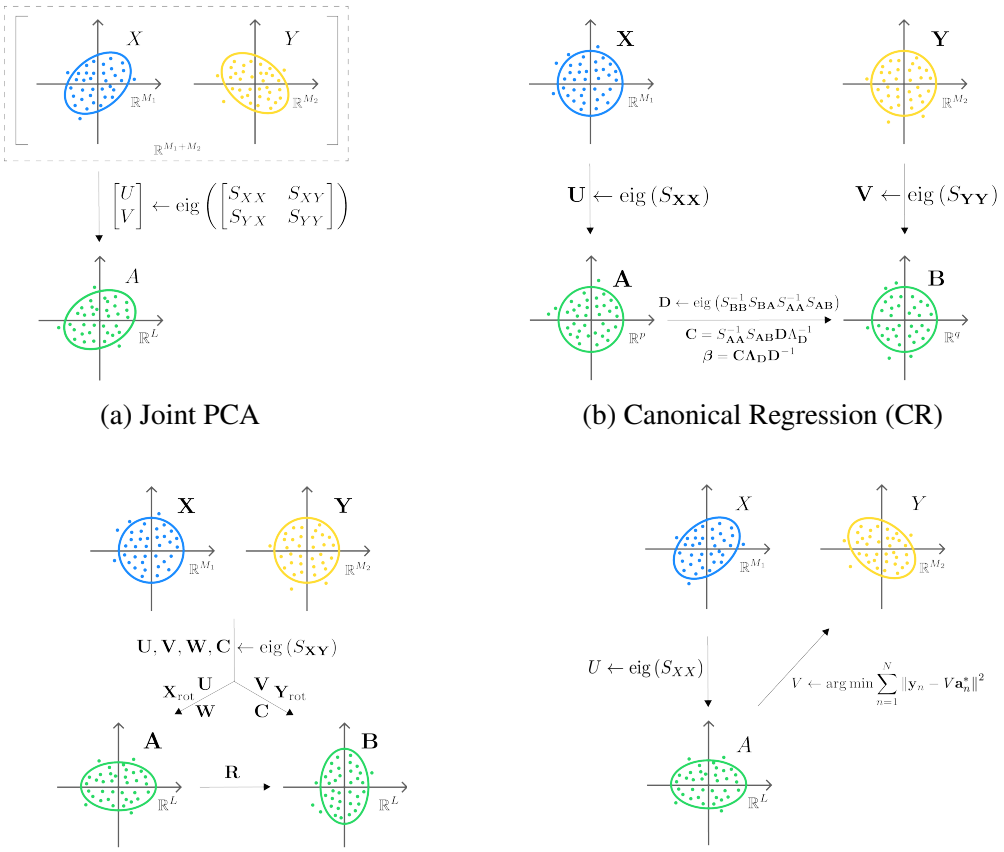
$$\hat{\mathbf{Y}}_{\text{test}} = \underbrace{\mathbf{V}\mathbf{R}}_{V \text{ in DP-PCA}} \underbrace{\left(\mathbf{W}^T\mathbf{U}\right)^{-1}\mathbf{W}^T}_{\mathbf{X}^T_{\text{rotations}}} \mathbf{X}_{\text{test}} = \boldsymbol{\beta}\mathbf{X}_{\text{test}} \tag{6.28}$$

6.3.2 Comparison on Correlation Analysis

Under the above framework, we can further highlight the differences and connections between the proposed Directionally Paired PCA and PLSR. One difference is that the loadings and scores and computed by optimizing different objective functions (i.e., reconstruction errors in Directionally Paired PCA and covariance in PLSR). Another connection is that we can consider $\mathbf{V}\mathbf{R}$ in PLSR as the equivalent paired basis V in Directionally Paired PCA as both of them are typically not orthogonal and map the expansion coefficients of X (i.e., A in Directionally Paired PCA and $\mathbf{A}$ in PLSR) to the unobservable data Y. In PLSR, the equivalent $V = \mathbf{U}\mathbf{R}$ is computed via two steps,[5] that is, covariance maximization for $\mathbf{V}$ and regression for $\mathbf{R}$. In the proposed Directionally Paired PCA, however, the paired basis V is computed one-shot by minimizing the reconstruction error on Y given the shared expansion coefficient A.

We can now compare the correlation analysis in relevant approaches in Figure 6.2. Joint PCA maximizes the full covariance matrix and obtains a concatenated basis, which is split into U, V (Figure 6.2a). Different from the original CCA whose goal is to maximize the correlation $\text{corr}(\mathbf{X}, \mathbf{Y})$ between the two sets of variables, the customized Canonical Re-

[5]The NIPALS [153] algorithms computes the matrices column by column in an iterative process.



(a) Joint PCA

(b) Canonical Regression (CR)

(c) Partial Least Square Regression (PLSR)

(d) proposed DP-PCA

Figure 6.2: Comparison on correlation analysis in related approaches.

gression (CR) first project both sets of standardized (i.e., subtracted by mean and divided by standard deviation) variables into the lower-dimensional subspace such that their variance is maximally captured. The regression coefficient is then computed by maximizing the covariance terms constructed by the paired standardized data after dimension reduction (Figure 6.2b). As for PLSR, the loadings $\mathbf{U}, \mathbf{V}$ and weights $\mathbf{W}, \mathbf{C}$ for transforms are estimated by maximizing the covariance cov($\mathbf{X}, \mathbf{Y}$), after which a regression analysis is performed between the scores (Figure 6.2c). Finally, in the proposed Directionally Paired PCA, the correlation between X and Y is maximized by sharing the expansion coefficient A in the lower-dimensional subspace (Figure 6.2d).

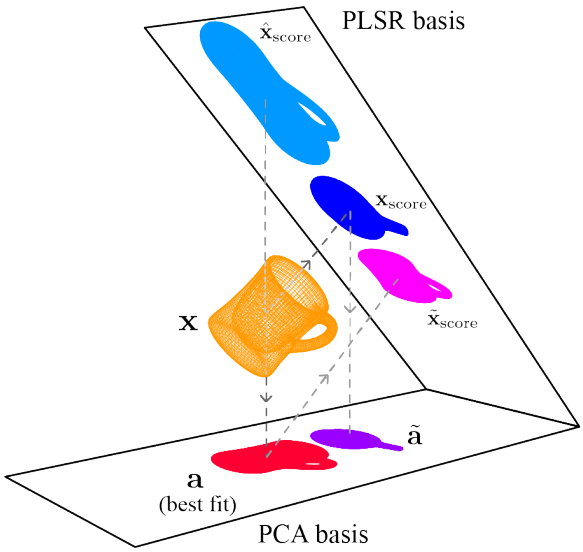


Figure 6.3: The "projection of a projection" when applying PLSR for inversion problems.

6.3.3 Specialty of Directionally Paired PCA for Inversion Problems

The uniqueness of the proposed Directionally Paired PCA lies in its special customization for inversion problems. On the contrary, those partial least-square methods (i.e., PLSR and CCA) apply only to the predication scenario, which assumes full access to the measurement X of the observable part. In practice, however, the low-dimensional representation of the test data A_{test} is often obtained by optimizing a loss function such that the representation maximally captures the variance of the raw data D_{test}. In other words, the representation always matches the PCA basis. Therefore, applying the PLS basis for solving the low-dimensional representation (i.e., expansion coefficient or score) would result in a situation of the "projection of a projection," which can be illustrated by Figure. 6.3.

Let us consider the high-dimensional measurement $\mathbf{x}$ of the observable part as a volume in the 3D space, which can be projected onto the PCA-basis plane or PLSR-basis plane, thus obtaining the low-dimensional representation $\mathbf{a}$ and $\mathbf{x}_{\text{score}}$, respectively. Under the PCA basis, $\mathbf{a}$ is the vector of expansion coefficients that maximally capture the variance (i.e., best fit) of the high-dimensional measurement $\mathbf{x}$. Under the PLSR basis, $\mathbf{x}_{\text{score}}$ is the best low-dimensional representation of the original measurement $\mathbf{x}$ that leads to the optimal

prediction of the unobservable part $\mathbf{y}$. At the test time of an inversion problem, the evolution of the low-dimensional representation (i.e., expansion coefficient) is driven by the effective projection on the PCA plane.

When PLS-basis is involved in inversion problems, two types of mismatch will exist: On the one hand, if we fit the coefficients $\mathbf{a}$ under the PCA basis but invert them using the PLSR basis, only the projection from $\mathbf{a}$ to the PLSR basis plane (i.e., $\tilde{\mathbf{x}}_{\text{score}}$) contributes to the inversion process. On the other hand, fitting and inverting the scores with the PLSR basis would lead to $\hat{\mathbf{x}}_{\text{score}}$ as its effective projection on the PCA basis is approximately equal to $\mathbf{a}$. In both cases, however, the low-dimensional scores $\tilde{\mathbf{x}}_{\text{score}}$ and $\hat{\mathbf{x}}_{\text{score}}$ do not match the loadings $\mathbf{U}$ and $\mathbf{VR}$ learned during training because $\mathbf{U}$ and $\mathbf{VR}$ are expected to map the true score $\mathbf{x}_{\text{score}}$ back to the high-dimensional space of observable measurements $\mathbf{x}$ and $\mathbf{y}$, respectively. Unfortunately, the required score $\mathbf{x}_{\text{score}}$ for PLSR is not reachable at test time. Should it be reachable, its projection on the PCA basis would become $\tilde{\mathbf{a}}$, which appears to be different from the actual $\mathbf{a}$. Unless we keep another pair of PCA bases with which we can reconstruct the high-dimensional measurements $\mathbf{x}$, PLS methods are not suitable for inversion problems.

6.4 Optimal Estimation for the Unobservable Part

Inspired by the strategy in Partial Least Square (PLS) methods for predicting the unobservable part Y from the observable part X, we can derive the optimal pair of bases U, V that directly minimizes the reconstruction error ε_Y for the unobservable part Y regardless of any correlation between the two variable sets X and Y. Henceforth, we refer to such estimation as the *optimal-Y* mode of the proposed Directionally Paired PCA. Starting with the loss function on U, V:

$$\varepsilon_Y(U, V) = \frac{1}{N} \sum_{n=1}^{N} \|\mathbf{y}_n - VU^T \mathbf{x}_n\|^2$$

$$= \frac{1}{N}\sum_{n=1}^{N} \mathbf{y}_n^T\mathbf{y}_n - 2\mathbf{x}_n^T UV^T\mathbf{y}_n + \mathbf{x}_n^T UV^T VU^T\mathbf{x}_n \tag{6.29}$$

Differentiating ε_Y with respect to U and V, we obtain:

$$\begin{aligned}\frac{\partial \varepsilon_Y}{\partial U} &= \frac{1}{N}\sum_{n=1}^{N} -2\mathbf{x}_n\left(V^T\mathbf{y}_n\right)^T + \left(2V^TVU^T\mathbf{x}_n\mathbf{x}_n^T\right)^T \\ &= -\frac{2}{N}\sum_{n=1}^{N} \mathbf{x}_n\mathbf{y}_n^TV - \mathbf{x}_n\mathbf{x}_n^TUV^TV \\ &= -\frac{2}{N}\left(XY^TV - XX^TUV^TV\right)\end{aligned} \tag{6.30}$$

and

$$\begin{aligned}\frac{\partial \varepsilon_Y}{\partial V} &= \frac{1}{N}\sum_{n=1}^{N} -2\mathbf{y}_n\mathbf{x}_n^TU + 2VU^T\mathbf{x}_n\mathbf{x}_n^TU \\ &= -\frac{2}{N}\left(YX^TU - VU^TXX^TU\right)\end{aligned} \tag{6.31}$$

Setting both matrix derivatives to zero yields the conditions:

$$\begin{cases} XY^TV = XX^TUV^TV \\ YX^TU = VU^TXX^TU \end{cases} \quad \text{or} \quad \begin{cases} XX^TU = XY^TV\left(V^TV\right)^{-1} \\ V = YX^TU\left(U^TXX^TU\right)^{-1} \end{cases} \tag{6.32}$$

Technically, the proposed Directionally Paired PCA can be described as least-square regression analysis in the low-dimensional (i.e., principal) subspace. The abbreviation of the principal least-square regression, however, coincides with the existing partial least-square regression (PLSR). Therefore, we name the method as Directionally Paired PCA, which also reflects the directional prediction in the subspace of principal components.

The following part of the section manages to solve the above conditions. If we substitute V in the top left equation of (6.32) with $YX^TU\left(U^TXX^TU\right)^{-1}$ as specified by the bottom

right equation (thus, $V^T = (U^T XX^T U)^{-1} U^T XY^T$), we obtain:

$$XY^T YX^T U \left(U^T XX^T U\right)^{-1} = XX^T U \left(U^T XX^T U\right)^{-1} U^T XY^T YX^T U \left(U^T XX^T U\right)^{-1} \tag{6.33}$$

which can be further simplified by right-multiplying $U^T XX^T U$ on both sides:

$$XY^T YX^T U = XX^T U \left(U^T XX^T U\right)^{-1} U^T XY^T YX^T U \tag{6.34}$$

Therefore, the optimal pair of bases U, V which minimizes the reconstruction error of the unobservable part Y can be computed by solving the above equation (6.34) for U and plugging the solution of U into the bottom right equation in (6.32) for V.

6.4.1 Direct Closed-form Solution

Notice that if we allow any solution (not necessarily orthogonal matrices) to equation (6.34), then it is easy to show by direct substitution that UW also solves the equation as long as W is any invertible $L \times L$ matrix.

$$\begin{aligned} XY^T YX^T UW &= XX^T U \left(U^T XX^T U\right)^{-1} U^T XY^T YX^T UW \\ &= XX^T UWW^{-1} \left(U^T XX^T U\right)^{-1} (W^T)^{-1} W^T U^T XY^T YX^T UW \\ &= XX^T UW \left(W^T U^T XX^T UW\right)^{-1} W^T U^T XY^T YX^T UW \end{aligned} \tag{6.35}$$

Thus, equation (6.34) depends only upon the column space of U. We may exploit such a property and seek a solution to the following simpler equation using any convenient choice of W:

$$XY^T YX^T U = XX^T UW \tag{6.36}$$

which has the exact same set of solutions.

When the dimension of measurements is much larger than the number of samples (i.e.,

$M \gg N$, common for inversion problems and 3D data), we can assume that X has full rank N, and the above equation can be further simplified for such an under-determined case as

$$Y^T Y X^T U = X^T U W \tag{6.37}$$

Now, if we perform an eigenvalue decomposition on the $N \times N$ matrix $Y^T Y$ and select L of the eigenvectors (with non-zero eigenvalues) to form the columns of an $N \times L$ matrix Z, and store their associated eigenvalues in an $L \times L$ diagonal matrix D, then we may write

$$Y^T Y Z = Z D \tag{6.38}$$

Finally, solving

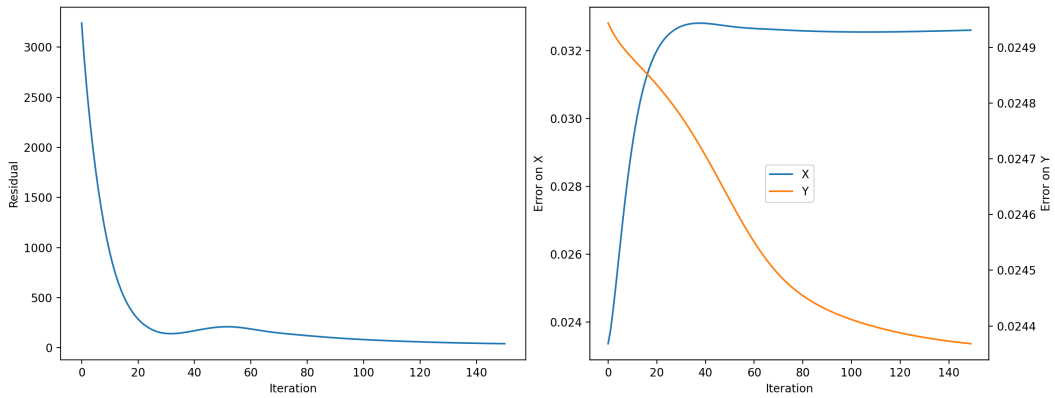
$$X^T U = Z \tag{6.39}$$

for U and setting $W = D$, we obtain a solution to equation (6.37) and hence a solution to equation (6.34).

Contrarily, when the number of training samples is larger than the dimension of data (i.e., $N > M$, common in statistical learning), we can select the largest eigenvalues of the matrix $Y^T Y$ and their associated eigenvectors for building the matrix Z. By setting the least-square solution to equation (6.39) as the initial value, we can compute the basis U with the following numerical solution via gradient descent.

6.4.2 Numerical Solution via Gradient Descent

An alternative to solving equation (6.34) for U is starting with the optimal choice of U for minimizing reconstruction errors for X (i.e., using standard PCA as earlier) or with the X_{rot} from PLSR, and combine the partial derivative results in equation (6.30) into the following



(a) residual (b) estimation errors

Figure 6.4: Illustration on the gradient descent evolution

gradient descent flow which improves the prediction performance for Y iteratively.

$$\frac{\mathrm{d}U}{\mathrm{d}t} = \overbrace{\frac{2}{N}\left(XY^T - XX^TU\underbrace{\left(U^TXX^TU\right)^{-1}U^TXY^T}_{V^T(U)}\right)\underbrace{YX^TU\left(U^TXX^TU\right)^{-1}}_{V(U)}}^{-\frac{\partial \varepsilon_Y}{\partial U}} \tag{6.40}$$

Figure 6.4 illustrates the gradient-descent evolution with the residual (Figure 6.4a) of equation (6.34) (i.e., the norm of the difference between the left- and right-hand sides) and estimation errors (Figure 6.4b) on X and Y. As we keep updating the basis U from the standard PCA basis following the gradient flow and computing its paired basis V with the bottom-right formula in equation (6.32), it gets closer to the solution of equation (6.34), which is reflected by the convergence of the residual at a low level. Consequently, the estimation error on Y decreases throughout the iterations at a cost that the errors on X become significantly larger.

6.4.3 Remarks on DP-PCA

Technically, the proposed DP-PCA can be described as a least-square regression analysis using a low-dimensional (i.e., principal) subspace. The abbreviation of the principal least-square regression, however, coincides with the existing partial least-square regression

(PLSR). Therefore, we name the method as DP-PCA, which also reflects the directional prediction in the subspace of principal components. The directionality of the proposed method is also reflected in the solutions. For (conditional) DP-PCA, we first solve the eigenvalue problem on XX^T in standard PCA on the observable part X, then minimize the least-square prediction error by setting the partial derivative of paired basis V to zero. For the unconditional ("optimal-Y" mode) DP-PCA, we begin by solving another eigenvalue problem on Y^TY (not YY^T, but their eigenvalues and eigenvectors are closely related), then solve a linear equation by minimizing the least-square error.

6.5 Evaluation on Dimension Reduction via Data Reconstruction

In this section, we evaluate the performance of relevant dimension reduction approaches via data-reconstruction experiments. We assume that better dimension reduction methods are more capable of capturing the principal components of the data and lead to lower reconstruction errors. Independent PCA provides the lower bound of the reconstruction error under the premise that both groups of coupled data X and Y are observable at all times. In reality, the paired data are available only during training, and we no longer have access to the unobservable variables Y at test time. Thus, the goal of the experiment with involved dimension-reduction methods is to both reconstruct the observable variables X and predict the unobservable variables Y using the bases U, V and expansion coefficients learned during training. Our simulation and experiment account for both prediction and inversion problems.

6.5.1 Experiment Design and Procedures

It should be highlighted that the purpose of the experiment is to evaluate the relevant algorithms in terms of dimension reduction rather than data reconstruction or prediction. At first glance, one may argue that the goal of the experiment is also achievable via autoencoders [161] (i.e., for reconstructing the observable variables X) and neural-network re-

gressors (i.e., for predicting the unobservable variables Y as multi-target regression). Those methods, however, fail to provide dimension reduction and correlation analysis in a similar manner as the involved algorithms do. In particular, correlation analysis is missing in auto-encoders, and neural-network regressors are not suitable for dimension reduction. Therefore, we consider the following approaches in our experiments: independent PCA[6] (Chapter 6.2.1), joint PCA[6] (Chapter 6.2.2), Partial Least Square Regression[7] (PLSR) [153], Canonical Regression[8] (CR) [156, 157], and the proposed Directionally Paired PCA (Chapter 6.3). Among the approaches, the independent PCA serves as the lower bound of reconstruction error, and both X, Y remain observable at all times. The Canonical Regression (CR) approach is a customized version of the Canonical Correlation Analysis (CCA), which supports predicting the values of Y from X.

Following the notation defined in Chapter 6.2, M_1 and M_2 denote the dimensions of observable (high-confidence) and unobservable (low-confidence) variables in a data sample, respectively. For each method, we compute the paired bases (i.e., loadings) U, V with the training set, reducing the dimensions of $X_{\text{train}}, Y_{\text{train}}$ from M_1, M_2 to L. In cases of a prediction problem, we apply the corresponding basis U or rotations $\mathbf{X}^T_{\text{rotation}}$ for dimension reduction to the observable test data[9] X_{test} and obtain the dimension-reduced data A_{test} with dimension L. As for the simulation of an inversion problem, the low-dimensional expansion coefficient A_{test} is always computed via dimension reduction with the basis U from standard PCA, which maximally captures the variance of X_{test}. The observable part of the reconstructed test data $\hat{X}_{\text{test}}$ is obtained by taking the inverse transform of the dimension reduction with its corresponding basis U or loadings $\mathbf{U}$.

The prediction of the unobservable part $\hat{Y}_{\text{test}}$ is handled differently in the involved approaches. For independent and joint PCA, the basis V characterizes a transformation be-

[6]Python implementation provided by scikit-learn at: `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`

[7]Python implementation provided by scikit-learn at: `https://scikit-learn.org/stable/modules/generated/sklearn.cross_decomposition.PLSRegression.html`

[8]R implementation at: `https://rdrr.io/github/jmhewitt/telefit/man/cca.predict.html`

[9]Applying the learned basis U again to X_{train} leads to expansion coefficient A in Chapter 6.2.

Table 6.1: Storage requirement on dimension reduction approaches for coupled data.

Method	Results to be stored after training
Joint PCA	$\bar{X}, \bar{Y}$: mean values of training data (with size M_1 and M_2); U, V: bases for X and Y (with size $M_1 \times L$ and $M_2 \times L$).
PLSR	$\bar{\mathbf{X}}, \bar{\mathbf{Y}}$: mean values of training data (with size M_1 and M_2); $\sigma_{\mathbf{X}}, \sigma_{\mathbf{Y}}$: standard deviations of training data (with size M_1 and M_2); $\mathbf{U}, \mathbf{X}_{\text{rotations}}$: loadings and rotations for $\mathbf{X}$ (both with size $M_1 \times L$); One of the following: (1) $\boldsymbol{\beta}$: regression coefficients for predicting $\mathbf{Y}$ (with size $M_2 \times M_1$) (2) $\mathbf{V}$: loadings for $\mathbf{Y}$ (with size $M_2 \times L$) and $\mathbf{R}$: relation matrix between $\mathbf{A}$ and $\mathbf{B}$ (with size $L \times L$)
CR	$\bar{\mathbf{X}}, \bar{\mathbf{Y}}$: mean values of training data (with size M_1 and M_2); $\sigma_{\mathbf{X}}, \sigma_{\mathbf{Y}}$: standard deviations of training data (with size M_1 and M_2); $\mathbf{U}, \mathbf{V}$: bases for $\mathbf{X}$ and $\mathbf{Y}$ (with size $M_1 \times L$ and $M_2 \times L$); $\bar{\mathbf{A}}, \bar{\mathbf{B}}$: mean values in the subspace (with size L); $\sigma_{\mathbf{A}}, \sigma_{\mathbf{B}}$: standard deviation in the subspace (with size L); $\boldsymbol{\beta}$: correlation coefficient between $\mathbf{A}$ and $\mathbf{B}$ (with size $L \times L$).
DP-PCA	same as those in the Joint PCA

tween the unobservable variables Y and their corresponding dimension-reduced expansion coefficients (i.e., scores) B. The only difference between independent and joint PCA is that we assume Y_{test} remains available for computing B_{test} in independent PCA whereas $A_{\text{test}} = B_{\text{test}}$ in joint PCA because only X_{test} is available. To predict the values of the unobservable part $\hat{Y}_{\text{test}}$, both independent PCA and joint PCA take the inverse transform of B_{test}. For PLSR, CR, and the proposed Directionally Paired PCA, the loadings V characterizes a transformation from the expansion coefficients A (i.e., scores) of the observable part to the unobservable data Y. To predict the values of the unobservable test data $\hat{Y}_{\text{test}}$, those three approaches apply a prediction transform to A_{test}. Based on the reconstructed and predicted values of the test data, we finally compute the mean squared error per element between $\{X_{\text{test}}, Y_{\text{test}}\}$ and $\{\hat{X}_{\text{test}}, \hat{Y}_{\text{test}}\}$.

Table 6.1 provides a list of the storage requirement for each approach to facilitate the reconstruction and prediction process. The data structures are consistent with the publicly available implementations discussed above. Compared with baseline methods PLSR and

CR, the proposed Directionally Paired PCA requires minimal storage that remains the same as joint PCA. In the following subsection, we introduce the benchmark datasets for our experiments and report execution time for those methods.

6.5.2 Benchmark Datasets and Execution Time

Datasets

We conduct experiments on three types of datasets: *synthetic*, *multi-target regression*, and *single-channel image* data. The details of those datasets elaborated as follows.

Synthetic data: We generate a data matrix D of size $(M_1 + M_2) \times N$, containing N data measurements from a multi-variate Gaussian distribution with random mean $\mu_{M_1+M_2}$ and random covariance matrix $\Sigma_{M_1+M_2}$. We then split the rows of D into the observable part X with size $M_1 \times N$ and unobservable part Y with size $M_2 \times N$. Thus, the correlation between the two parts are established via the covariance matrix $\Sigma_{M_1+M_2}$. By keeping 70% samples for training and the rest for testing, the N data samples are further divided into training set $\{X_{\text{train}}, Y_{\text{train}}\}$ and test set $\{X_{\text{test}}, Y_{\text{test}}\}$. Following the procedures illustrated in the previous subsection, the training set is used for computing bases U, V and other required results listed in Table 6.1, whereas the test set is reserved for computing reconstruction/inversion (for X_{test}) and prediction (for Y_{test}) errors.

Multi-target regression data: As discussed at the beginning of Chapter 6.5.1, predicting the values of the unobservable variables Y can be formulated as a multi-target regression problem. In multi-target regression datasets, the observable variables X are called "features" while the unobservable variables Y are considered "targets." Among all 18 datasets in [162], we select 4 of them that satisfy the following two conditions. (1) the dimensions of both X and Y are larger than 10 so that there is room for varying the dimension L of the subspace, and (2) no missing values exist in the data.

Single-channel image data: We further repeat the experiments on MNIST [114], which are real datasets with larger dimensions than those of the multi-target regression

datasets. Pixels of each image are split into two halves as observable and unobservable either (1) according to the sequence of indices (i.e., sequential split) or (2) randomly yet consistently across images (i.e., random split).

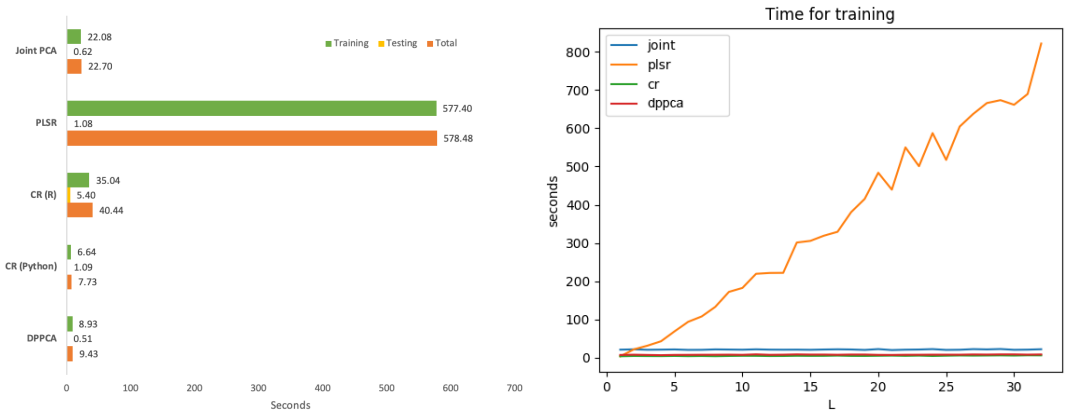
Execution time

We report the execution time of each method for 100 runs on the synthetic dataset in Figure 6.5. The execution time is benchmarked on a Ubuntu 16.04 Desktop with 8-core Intel Core i7-6700K CPU @ 4.00GHz and 16GB DDR4 RAM @ 2133MHz. In a strict sense, the reported execution time does not necessarily demonstrate the time complexity of the approaches because they are not optimized uniformly. Instead, the chart in Figure 6.5a reflects the experience with popular implementations that are publicly available. According to the chart, the required training time for PLSR is substantially longer than others. In addition, as illustrated by Figure 6.5b, the training time in PLSR also increases significantly as the budget (i.e., the dimension of the target subspaces) increases. As for the testing time, all approaches have testing time fluctuated within a small range. We also compare the execution time for CR between the original R implementation and our translated version[10] in Python, and find out that the Python version is about 5 times faster. In sum, the proposed Directionally Paired PCA is relatively faster than its competitors provided by popular open-source implementations.

6.5.3 Simulation of the Inversion Problem

In this subsection, we simulate the inversion problem and demonstrate the "projection of a projection" effect illustrated in Figure 6.3. Following the procedures described in Chapter 6.5.1, we initiate the test-time inference by performing dimension reduction on test data X_{test} with the standard-PCA basis U that are learned from training data X_{train}. For an inversion problem, we have no direct access to the high-dimensional measurement X itself;

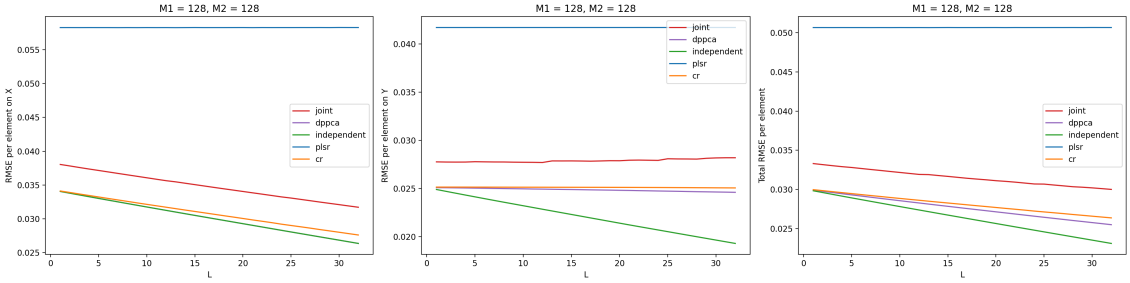
[10] `https://gist.github.com/thelittlekid/89630241f5b90a838a7b583a5836d350`



(a) Execution time for 100 runs ($M_1 = M_2 = 128, L = 32$)

(b) Training time for 100 runs ($M_1 = M_2 = 128, L = 1$ to 32)

Figure 6.5: Comparison on execution time on the synthetic dataset.



(a) Error of inversion on observable X_{test}

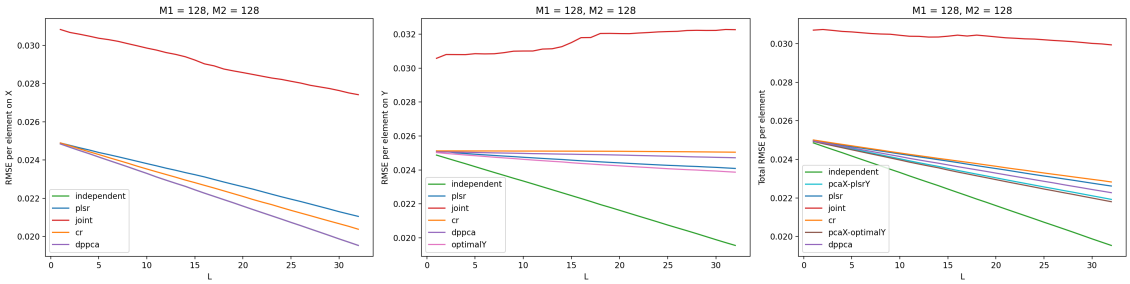
(b) Prediction error on unobservable Y_{test}

(c) Total reconstruction error

Figure 6.6: Evaluation on dimension reduction via simulated inversion and prediction of coupled synthetic data. $N = 10^4$, $M_1 = M_2 = 128$, $L = 1$ to 32. Horizontal axis: dimension L of the target subspace (i.e., budget); vertical axis: inversion/prediction error.

instead, we can estimate its low-dimensional representation A_{test} by optimizing an energy function whose objective is in line with capturing the maximum variance of each test sample. Thus, the energy optimization process can be simulated using dimension reduction on X_{test} with standard PCA. Once we obtain the low-dimensional representation A_{test}, we can then predict the unobservable measurement $\hat{Y}_{\text{test}}$ following the standard routine of the respective algorithms.

Figure 6.6 shows the root-mean-square error per element for the inverted and predicted signals. The proposed Directionally Paired PCA outperforms competing algorithms and achieves lower errors that are closest to the lower bound from standard independent PCA.



(a) Reconstruction error on observable X_{test}

(b) Prediction error on unobservable Y_{test}

(c) Total reconstruction error

Figure 6.7: Evaluation on dimension reduction via data reconstruction and prediction of coupled synthetic data. $N = 10^4$, $M_1 = M_2 = 128$, $L = 1$ to 32. Horizontal axis: dimension L of the target subspace (i.e., budget); vertical axis: reconstruction/prediction error.

As a consequence of the mismatch between scores and loadings, errors in PLSR are significantly higher than others. On the other two real datasets, the errors are not even on the same scale. In reality, inversion problems often occur in 3D segmentation and reconstruction problems. For the demonstration of the proposed Directionally Paired PCA in those applications, we refer readers to the parallel research in our lab. For the rest of this section, we shift our focus from inversion problems to prediction problems often encountered in statistical analysis.

6.5.4 Experiments in the Prediction Scenario

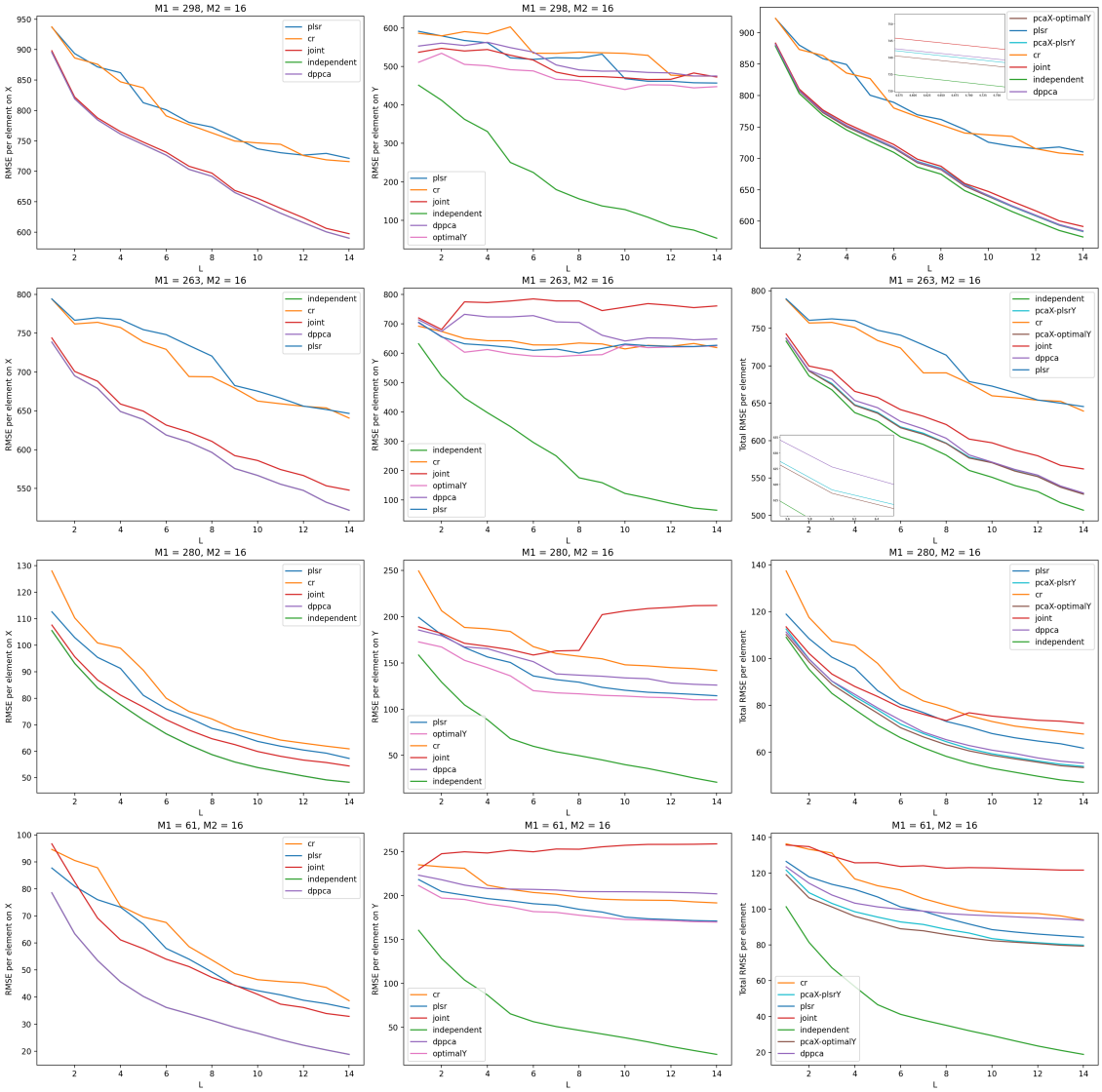
Despite its origin from a use case of the inversion problem, Directionally Paired PCA is also applicable to prediction problems and beats the existing linear approaches. In this section, we switch to the prediction scenario by assuming that the high-dimensional measurements X of the observable part are accessible at test time. Correspondingly, the reconstructed signal of the observable measurement $\hat{X}_{\text{test}}$ is calculated by taking the (forward) transform and then the inverse transform of a dimension reduction approach. The prediction of the unobservable part Y_{test} remains the same as that described in Chapter 6.5.1.

Figure 6.7 illustrates the reconstruction and prediction errors on the synthetic multivariate Gaussian data using involved approaches. During training, joint PCA shares the

budget on the dimension of the subspace L and between the observable part X and unobservable part Y. Consequently, the principal components of X are no longer optimal, and the effort spent on Y is wasted because Y becomes unobservable at test time. Such results motivate us to design Directionally Paired PCA that both accounts for the correlation between X and Y and optimally preserves the observable part X. Thus, the reconstruction error on the observable part X remains optimal for Directionally Paired PCA and meets the lower bound given by the independent PCA. As the objective of CCA (i.e., CR) is to maximize the correlation between X and Y, neither bases are optimal in preserving information for data reconstruction. By design, PLSR aims at predicting the unobservable part Y by leveraging the covariance between the two variable sets X and Y, leading to more precise predictions on Y yet larger distortion on X. If we disregard the correlation and target at the best prediction with the optimal-Y mode in Directionally Paired PCA, we can further push higher the accuracy on the unobservable Y at the cost of substantial reconstruction errors on the observable X, which are orders of magnitude larger and not suitable for plotting with other methods in Figure 6.7a.

Overall, when combining observable and unobservable variables (Figure 6.7c), the proposed Directionally Paired PCA achieves the lowest errors under the circumstance that only one pair of bases are allowed. With the additional budget for the PCA basis of the observable part X, we can fully enjoy the benefits from the optimal prediction on Y using a combined method: applying standard independent PCA for the observable part X and Directionally Paired PCA in mode optimal-Y for the unobservable part Y (i.e., "pcaX-optimalY"). Theoretically, such a combination is the best possible linear model for the estimation of coupled data.

Figure 6.8 shows the reconstruction and prediction errors on the 6 selected multi-target regression datasets. Despite slight differences, results on real data exhibit a similar pattern as those on the synthetic data in Figure 6.7. On real data, the simplest joint PCA does not necessarily lead to the largest errors, even though the proposed Directionally Paired

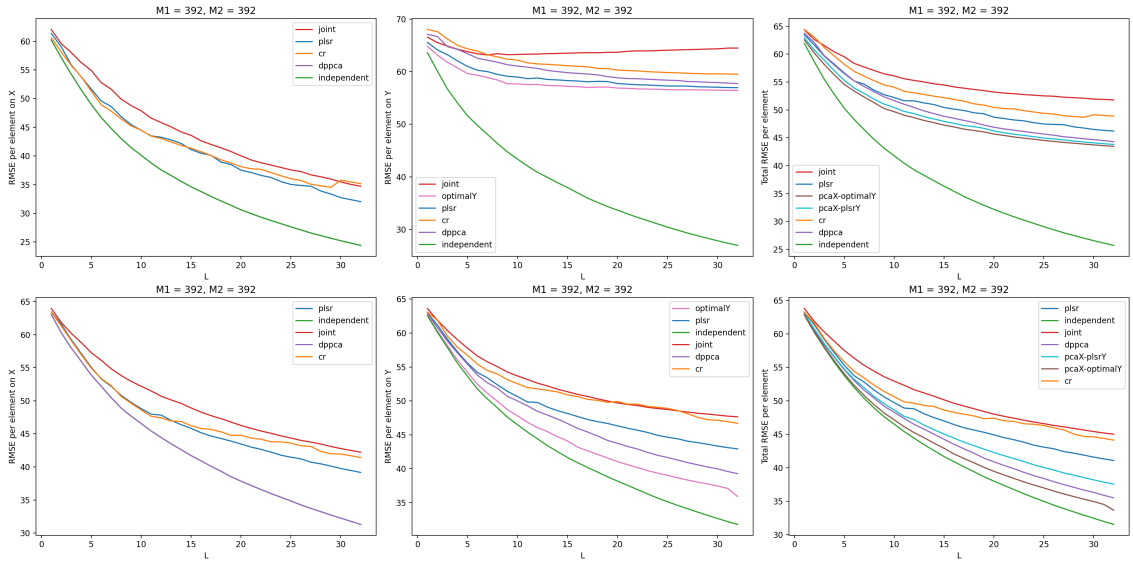


(a) Reconstruction error on observable X_{test}

(b) Prediction error on unobservable Y_{test}

(c) Total reconstruction error

Figure 6.8: Evaluation on dimension reduction via data reconstruction and prediction of real multi-target regression datasets [163, 164]: (top to bottom) oes10, oes97, scm1d, scm20d, and wq. Horizontal axis: dimension L of the target subspace (i.e., budget); vertical axis: reconstruction/prediction error.



(a) Reconstruction error on observable X_{test}

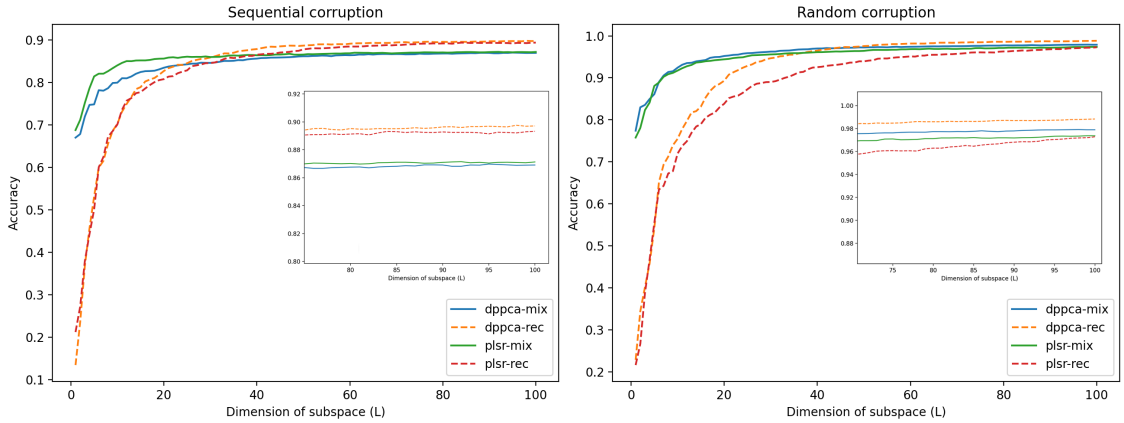
(b) Prediction error on unobservable Y_{test}

(c) Total reconstruction error

Figure 6.9: Evaluation on dimension reduction via data reconstruction and prediction of MNIST. $L = 1$ to 32, $M_1 = M_2 = 392$ (equal split of variables). Top row: sequential split; bottom row: random split.

PCA consistently outperforms it. The pursue of maximum correlation from CR leads to sub-optimal reconstruction on both the observable and unobservable variables. With PLSR and the optimal Y mode of Directionally Paired PCA, one may achieve lower prediction errors on the unobservable part Y. When it comes to the total reconstruction error, however, the proposed Directionally Paired PCA beats others with single pair of bases most of the time (with a few exceptions on the 4^{th} row scm20d dataset with large budgets L, but we are more interested in lower-budget scenarios for dimension reduction). The combined method of "pcaX-optimalY" remains to be the best linear solution on real data. In Figure 6.9, we further repeat the experiments on MNIST [117], which is a real dataset with larger input dimension. Results on MNIST agree with the previous ones; when pixels are randomly split, the proposed Directionally Paired PCA even obtains the lowest errors on both the observable and unobservable variables (bottom row of Figure 6.9).

Finally, to demonstrate the effectiveness in capturing principal components of the signals, we classify the reconstructed images using a pretrained classifier trained on clean,



(a) sequential missing (b) random-consistent missing

Figure 6.10: Classification accuracy on partially observable and noisy MNIST after reconstruction and prediction: half of the pixels are missing and the other half noisy at test time.

complete samples. The experiment corresponds to a use case in which half of the pixels (i.e., Y_{test}) are unobservable at test time. Similar to Figure 6.9, the pixels either missing sequentially (i.e., the bottom half of the image) or random yet consistently across images. Moreover, the other observable half (i.e., X_{test}) is interfered by zero-mean Gaussian noise with $\sigma = 0.3$ (in an intensity scale of $[0, 1]$). The input images to the classifier are built with two different modes: mixture or reconstruction. In the mixture mode, we combine the available X_{test} with the predicted $\hat{Y}_{\text{test}}$ whereas in the reconstruction mode, we integrate the reconstructed signal for the observable part $\hat{X}_{\text{test}}$ and the predicted $\hat{Y}_{\text{test}}$. Models for linear reconstruction and prediction are trained on 10,000 randomly selected samples from the original training set. In contrast to retraining the classifier, the ground-truth labels are no longer required for training those linear models.

Figure 6.10 illustrate the classification accuracy on reconstructed signals of MNIST images. Under low budgets (i.e., small L), mixing the predicted unobservable $\hat{Y}_{\text{test}}$ from PLSR with observable X_{test} leads to higher accuracy. As the budget becomes sufficiently large, replacing the observable X_{test} with the reconstructed version $\hat{X}_{\text{test}}$ results in higher accuracy. It is worth mentioning that even though that (conditional) DP-PCA obtain higher prediction errors on $\hat{Y}_{\text{test}}$ than PLSR, it achieves lower errors both on $\hat{X}_{\text{test}}$ and in total, thus

leading to highest classification accuracy under larger budget.

6.5.5 Result Analysis

After evaluating the relevant linear models in both inversion and prediction problems, we now analyze the results in terms of degrees of freedom (i.e., budgets) in the optimization process. The degrees of freedom characterizes the number of free variables to be optimized in each method. For independent and joint PCA as well as the proposed Directionally Paired PCA, the budget equals the total number of variables in U and V, that is, $(M_1 + M_2) \times L$. When it comes to PLSR and CR, an additional budget of L^2 is introduced to learn the mapping between the paired data after dimension reduction. In an ideal case such as two independent PCAs, the degrees of freedom are proportionally split between the observable part X and unobservable part Y at a ratio of $M_1 : M_2$, and each part of the budget is optimally spent to minimize the reconstruction errors, respectively. When one set of variables Y becomes unobservable, the corresponding part of the budget is dissipated while the other part for X is utilized in a sub-optimal manner also to take account of the correlation between the two sets. The proposed Directionally Paired PCA ensures that the budget spent on the observable part is utilized optimally such that it maximally captures the variance and minimizes the reconstruction error. The other part of the budget on unobservable Y, moreover, is consumed in the best possible fashion for minimizing the reconstruction error given the shared expansion coefficients. As far as the optimal Y mode, all budgets are allocated to predict the unobservable part Y. In practice, we may assume that M_1 and M_2 are much larger than L. Thus, the majority of the degrees of freedom (i.e., $(M_1 + M_2) \times L$) in PLSR is allocated to maximize the covariance between the two sets. The method does not explicitly capture variance or minimize reconstruction errors for the observable part X, sometimes leading to higher reconstruction errors on the observable part. On the contrary, with a better correlation and extra budget of L^2 on regression, it tends to predict the values of the unobservable part Y better. In Canonical Regression, the most critical L^2 degrees of

freedom are reserved for correlation analysis in the sub-spaces instead of starting from the original high-dimensional data, leading to worse predictions than PLSR. Although dividing the inputs by their standard deviation appears to be a valid strategy for data visualization and regression analysis, it is less desirable to minimize the reconstruction errors.

In conclusion, we make the following statements: When estimating coupled yet partially observable data in an inversion problem, the proposed Directionally Paired PCA is the only feasible linear model. When it comes to a prediction scenario, one can achieve the lowest overall reconstruction errors by applying standard PCA for the observable part X and the optimal Y mode of the proposed Directionally Paired PCA for the unobservable part Y. Such a combined approach, however, requires two separate sets of bases, resulting in longer computation time and larger storage. In cases when the unobservable part Y is no more critical than the observable part X, the proposed Directionally Paired PCA approach can achieve the lowest total error in estimation with a single pair of bases at a fast speed.

6.6 Summary

In this chapter, we present a novel Asymmetric Principal Component Analysis for the estimation of coupled data. As the dimension of useful features in data is generally much smaller than that of the data themselves, dimension reduction methods such as PCA are often conducted in data modeling and analysis for single datasets. When analyzing coupled datasets, joint PCA split the budget between the two but fails in cases where one of them is unobservable. Previous algorithms such as CCA and PLS methods maximize the correlation between the two datasets, yet leading to sub-optimal solutions for maximizing variance and minimizing reconstruction errors on each of them. The proposed Directionally Paired PCA combines the strength of both PCA and correlation analysis by optimally expressing the variability of the observable part, meanwhile maximally capturing the correlation between the two by sharing the expansion coefficients. It is explicitly designed for minimizing the reconstruction and prediction errors to the principal low-dimensional

space. In summary, the author's contributions in the collaborative work are listed below.

- A comprehensive analysis and elaboration on the proposed Directionally Paired PCA and its connection to related algorithms, including joint PCA, CCA, and PLSR.
- A demonstration on the effectiveness of the proposed Directionally Paired PCA and its superiority over existing approaches in terms of the estimation of coupled data when one of them becomes unobservable at test time.

At the end of the chapter, it is worth stating that the proposed Directionally Paired PCA should be considered a general framework and could be applied to various problems. For more examples of the application in myocardial segmentation, we refer readers to the research of Navdeep Dahiya in our Laboratory of Computational Computer Vision.

CHAPTER 7
CONCLUSION AND DISCUSSION

In this dissertation, we start by observing various robustness issues in machine learning, which suggest a gap between human and machine perception. To address those issues, we propose a conceptual hypothetical image-space model and a topological view of AI-knowledge based on which we further provide a unified explanation. In the following chapters, we design experiments to validate our hypothetical model as well as defined concepts of AI-knowledge. Throughout the experiments in two concrete applications, single-label classification and photo stylization, we demonstrate the benefits of adopting the philosophy from the proposed viewpoint.

With the hypothetical model validated by experiments, we can reexamine the three significant differences between the AI and our brains, which were pointed out at the beginning of the dissertation. In a classification task, a classifier learns to partition the entire input image space. Humans are never limited by a fixed number of categories when learning concepts and objects, and we can easily connect relevant samples in the input space even though they are not path-connected. On the contrary, modern neural networks are restricted by the number of categories and tend to group the samples into path-connected regions, creating uncertain bridges between them, which give birth to robustness issues such as adversarial examples. In a lifelong learning scenario, our brain learns better features by reviewing and comparing samples from a massive number of categories. In contrast, an AI classifier focuses on differentiating the categories in the current task with the most efficient descriptors. When those categories disappear in later tasks, it becomes unnecessary to remember the partition for them as they become irrelevant to minimizing the error from current training samples.

According to Rich Sutton, the bitter lesson[1] from past AI research suggests that scalable learning methods often supersede human-centric approaches that take advantage of human knowledge and understanding. In this dissertation, we ultimately put ourselves in the machine's position and define fundamental concepts following its perception process. Those concepts may later serve as basic ingredients for building scalable learning algorithms. Compared with AI, human intelligence are much more capable at the strategic level, especially in designing fundamental elements in learning frameworks. In single-label classification, we introduce a new degree of freedom on categories to the learning process by revealing hidden information that used to be overlooked in the past. In photographic style recognition, we shift from hand-crafted features to feature representation learned from constructed bases categories. Both strategies have the potential for scaling with more computation resources and data.

At the end of the dissertation, we discuss possible directions for future research.

- The variational-calculus view of machine learning implies that there might exist an optimal training set for each test sample. Ideally, when future computation resource is sufficient, the inference process may turn into a "many-shot" learning problem: for each test samples, the agent first selects an optimal training set and trains a classifier on it. Such a workflow, however, may lead to a "chicken or the egg" causality dilemma because a good representation is required at the sample selection stage.

- The extra categories for building "poorly justified true beliefs" or "false beliefs" are not necessarily required to be random noise or categories within the same dataset. It might be possible that reference data from other datasets can enhance classification accuracy on the target dataset. Moreover, the composition of categories may also be automated in the future.

- The possibility of inflating accuracy for defenses against adversarial attacks and the

[1] `http://incompleteideas.net/IncIdeas/BitterLesson.html`

large variance of adversarial robustness among categories and samples raises a request for more comprehensive evaluation metrics for measuring adversarial robustness.

- The gap between HSV and other color spaces reveals the limitation of neural networks in approximating certain functions, which calls for improvement at universal approximation.

- The preset classification benchmark demonstrates an example for learning abstract concepts beyond objects and contents with neural networks, which leaves open possibilities for future work on building large-scale recognition challenges regarding abstract concepts under control. For preset classification, scaling the categories up to hundreds or thousands may lead to a deeper comprehension of the photographic styles.

- For photographic style recognition, instead of formulating it as a partition of the image space, a vector-field formulation with reference points may lead to the better measurement for positional-variant artistic styles.

Appendices

APPENDIX A

THE SCALE-SPACE EFFECT OF NATURAL-IMAGE SPACES

This section contains the reasoning process for the argument that higher-dimensional natural-image spaces are more sparse than lower-dimensional ones. Let d denote the dimension (product of height, width, and the number of channels) of an image space X with a lower dimension. We assume that among all 256^d images in that space, N of them appears to be natural. We scale up the size (i.e., both height and width) of images by a factor of n, leading to a higher-dimensional image space X' with dimension n^2d. Assuming that the number of natural images in X' is N', we argue that X' is sparser compared to X, that is $\frac{N}{256^d} > \frac{N'}{256^{n^2d}}$.

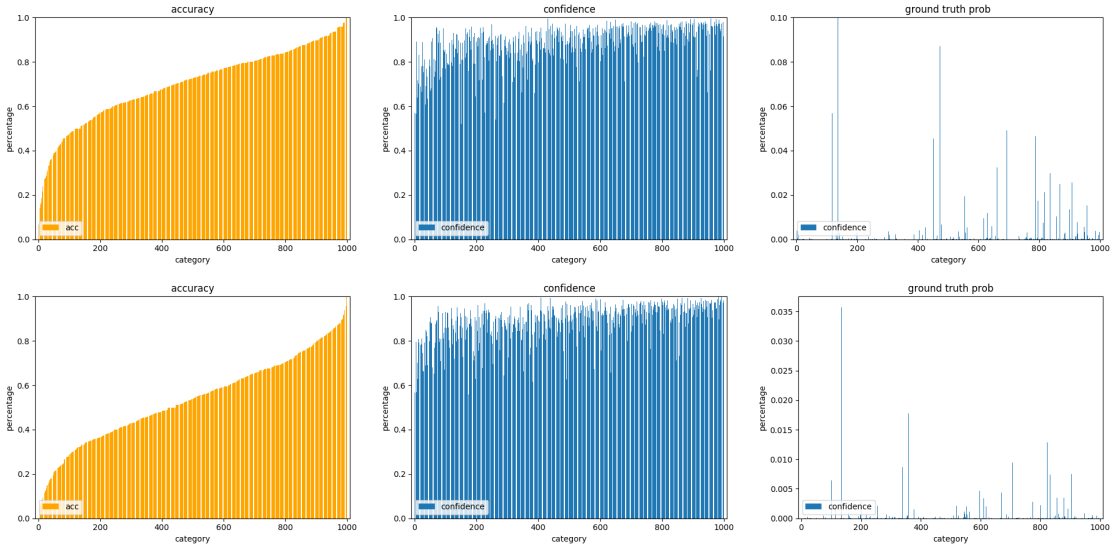
Reasoning by contradiction: Assume X' is not more sparser than X (i.e., $\frac{N}{256^d} \leq \frac{N'}{256^{n^2d}}$), we have $N' \geq 256^{(n^2-1)d}N$. The inequality implies that on average, for each lower-resolution image $I \in X$, there are at least $256^{(n^2-1)d}$ different higher-resolution images $I' \in X'$ which are derived from the center image I (e.g., via super-resolution followed by small modifications). We further assume that for each natural image I' in the higher-dimensional space X', the down-sampled version I in the lower-dimensional space should also appear to be natural because a discriminator (e.g., human eyes) are more sensitive to flaws in the higher-dimensional space. Then, we compute the average freedom of intensity-change per pixel in order to meet the minimum requirement:

$$x^{n^2d} = 256^{(n^2-1)d} \quad \Longrightarrow x = 256^{\frac{n^2-1}{n^2}} \tag{A.1}$$

The above result suggests that in order to keep the higher-dimensional image space X' as sparse as the lower-dimensional space X, a natural image $I' \in X'$ should remain natural when all pixels take arbitrary perturbations at a range of $[-\frac{x}{2}, \frac{x}{2}]$. Such a requirement will not be satisfied because the monotonically increasing $x = 64$ at $n = 2$.

APPENDIX B

PROPERTIES NOT CORRELATED TO THE CATEGORICAL VARIANCE OF ADVERSARIAL ROBUSTNESS



(a) accuracy (b) confidence (c) ground-truth probability

Figure B.1: Distribution of (a) sorted categorical accuracy on smoothed (with anisotropic diffusion) adversarial examples, (b) categorical confidence on unattacked samples, and (c) categorical probability on the ground-truth category for adversarial examples. Top/bottom row: adversarial examples are generated from PGD ($\epsilon = .01$ /$\epsilon = .05$). The indices of categories for graphs within each row are matched.

Two factors seem to be correlated to the categorical variance of adversarial robustness described in Section 4.4.4. One is that categories with higher confidences on clean images might be harder to attack, which potentially leads to higher accuracy after the defense. Another conjecture is that adversarial examples with a relatively higher probability assigned to the ground-truth category might be easier to defend. Figure B.1 disproves the assumptions, as categorical accuracy on smoothed adversarial examples increases (B.1a), there is no obvious trend in corresponding confidences on unattacked images (B.1b) or probability on the ground-truth category for adversarial examples (B.1c).

APPENDIX C

TRANSFORMATION FORMULAS AMONG COLOR SPACES

RGB and HSV

To convert integer values $R, G, B \in [0, 255]$ to $H \in [0, 360)$ and $S, V \in [0, 1]$:

$$R' = R/255, \quad G' = G/255, \quad B' = B/255$$

$$V = C_{\max} = \max(R', G', B'), \quad C_{\min} = \min(R', G', B'), \quad \Delta = C_{\max} - C_{\min}$$

$$H = \begin{cases} 0, & \Delta = 0 \\ 60^\circ \times (\frac{G'-B'}{\Delta} \mod 6), & C_{\max} = R' \\ 60^\circ \times (\frac{B'-R'}{\Delta} + 2), & C_{\max} = G' \\ 60^\circ \times (\frac{R'-G'}{\Delta} + 4), & C_{\max} = B' \end{cases}, \quad S = \begin{cases} 0, & C_{max} = 0 \\ \frac{\Delta}{C_{max}}, & C_{max} \neq 0 \end{cases}$$

To convert $H \in [0, 360)$ and $S, V \in [0, 1]$ to integer values $R, G, B \in [0, 255]$:

$$C = V \times S, \quad X = C \times (1 - |(H/60^\circ) \bmod 2 - 1|), \quad m = V - C$$

$$(R', G, B') = \begin{cases} (C, X, 0), & 0^\circ \leq H < 60^\circ \\ (X, C, 0), & 60^\circ \leq H < 120^\circ \\ (0, C, X), & 120^\circ \leq H < 180^\circ \\ (C, X, C), & 180^\circ \leq H < 240^\circ \\ (X, 0, C), & 240^\circ \leq H < 300^\circ \\ (C, 0, X), & 300^\circ \leq H < 360^\circ \end{cases}$$

$$(R, G, B) = ((R' + m) \cdot 255, (G' + m) \cdot 255, (B' + m) \cdot 255)$$

RGB to XYZ

We first define a pair of helper functions f and f^{-1}:

$$f(C) = \begin{cases} \left(\frac{C/255+.055}{1.055}\right)^{2.4}, & \frac{C}{255} > .4045 \\ \frac{C/255}{12.92}, & \text{otherwise} \end{cases}, \quad f^{-1}(C') = \begin{cases} 1.055 \cdot C'^{\frac{1}{2.4}} - .055, & C' > .0031308 \\ 12.92 \cdot C', & \text{otherwise} \end{cases}$$

To convert integer values $R, G, B \in [0, 255]$ to $X \in [0, .95047], Y \in [0, 1], Z \in [0, 1.08883]$. Apply the function f to each single channel R, G, B: $(R', G', B') = (f(R), f(G), f(B))$. Then, convert the intermediate results to the target space:

$$X = .4124 \cdot R' + .3576 \cdot G' + .1805 \cdot B'$$

$$Y = .2126 \cdot R' + .7152 \cdot G' + .0722 \cdot B'$$

$$Z = .0193 \cdot R' + .1192 \cdot G' + .9505 \cdot B'$$

To convert $X \in [0, .95047], Y \in [0, 1], Z \in [0, 1.08883]$ back to integer values $R, G, B \in [0, 255]$, we start with:

$$R' = 3.2406 \cdot X' - 1.5372 \cdot Y' - .4986 \cdot Z'$$

$$G' = .9689 \cdot X' + 1.8758 \cdot Y' + .0415 \cdot Z'$$

$$B' = .0557 \cdot X' - .2040 \cdot Y' + 1.0570 \cdot Z'$$

Finally, we obtain $(R, G, B) = 255 \cdot \left(f^{-1}(R'), f^{-1}(G'), f^{-1}(B')\right)$.

XYZ to Lab

We begin by defining a pair of helper functions f and f^{-1}.

$$f(C) = \begin{cases} C^{\frac{1}{3}}, & C > .008856 \\ 7.787 \cdot C + \frac{16}{116}, & \text{otherwise} \end{cases}, \quad f^{-1}(C') = \begin{cases} C'^3, & C'^3 > .008856 \\ \frac{(C'-16)/116}{7.787}, & \text{otherwise} \end{cases}$$

The reference white point is defined as $(X_n, Y_n, Z_n) = (95.0489, 100, 108.884)$.

To convert $X \in [0, .95047], Y \in [0, 1], Z \in [0, 1.08883]$ to $L \in [0, 100]$ and $a, b \in [-110, 110]$.

$$(X', Y', Z') = f(X/X_n,\ Y/Y_n,\ Z/Z_n) = f(X/95.0489,\ Y/100,\ Z/108.884)$$

$$(L, a, b) = ((116 \cdot Y') - 16,\ 500 \cdot (X' - Y'),\ 200 \cdot (Y' - Z'))$$

To convert $L \in [0, 100]$ and $a, b \in [-110, 110]$ to $X \in [0, .95047], Y \in [0, 1], Z \in [0, 1.08883]$.

$$L' = (L + 16)/116, \quad a' = a/500 + L', \quad b' = L' - b/200$$

$$(X', Y', Z') = (f^{-1}(a'),\ f^{-1}(L'),\ f^{-1}(b'))$$

$$(X,\ Y,\ Z) = (X' \cdot X_n,\ Y' \cdot Y_n,\ Z' \cdot Z_n) = (X' \cdot 95.0489,\ Y' \cdot 100,\ Z' \cdot 108.884)$$

RGB to YCbCr

To convert integer values $R, G, B \in [0, 255]$ to $Y \in [16, 235]$ and $C_b, C_r \in [16, 240]$:

$$Y = 16 + (65.481 \cdot R + 128.553 \cdot G + 24.966 \cdot B)$$

$$C_b = 128 + (-37.797 \cdot R - 74.203 \cdot G + 112.0 \cdot B)$$

$$C_r = 128 + (112.0 \cdot R + 93.786 \cdot G - 18.214 \cdot B)$$

The inverse transform is given by:

$$
\begin{aligned}
R &= \frac{255}{219} \cdot (Y - 16) &&&&&& + \frac{255}{224} \cdot 1.402 \cdot (C_r - 128) \\
G &= \frac{255}{219} \cdot (Y - 16) && - \frac{255}{224} \cdot 1.772 \cdot \frac{.114}{.587} \cdot (C_b - 128) && - \frac{255}{224} \cdot 1.402 \cdot \frac{.299}{.587} \cdot (C_r - 128) \\
B &= \frac{255}{219} \cdot (Y - 16) &&&& + \frac{255}{224} \cdot 1.772 \cdot (C_b - 128)
\end{aligned}
$$

RGB to HED

To convert integer values $R, G, B \in [0, 255]$ to H, E, D:

$$R' = \ln\left(\frac{R}{255} + 2\right), \quad G' = \ln\left(\frac{G}{255} + 2\right), \quad B' = \ln\left(\frac{B}{255} + 2\right)$$

$$H = 1.87798 \cdot R' - 1.00768 \cdot G' - .55612 \cdot B'$$

$$E = -.0691 \cdot R' + 1.13473 \cdot G' - .13552 \cdot B'$$

$$D = -.60191 \cdot R' - .48041 \cdot G' + 1.57359 \cdot B'$$

To convert H, E, D back to $R, G, B \in [0, 255]$:

$$R' = .65 \cdot H + .70 \cdot E + .29 \cdot D$$

$$G' = .07 \cdot H + .99 \cdot E + .11 \cdot D$$

$$B' = .27 \cdot H + .57 \cdot E + .78 \cdot D$$

$$R = 255 \cdot (e^{R'} - 2), \quad G = 255 \cdot (e^{G'} - 2), \quad B = 255 \cdot (e^{B'} - 2)$$

RGB to YUV

To convert integer values $R, G, B \in [0, 255]$ to $Y \in [0, 1]$, $U \in [-.436, .436]$, $V \in [-.615, .615]$:

$$R' = R/255, \quad G' = G/255, \quad B' = B/255$$

$$Y = .299 \cdot R' + .587 \cdot G' + .114 \cdot B'$$

$$U = -.14714119 \cdot R' - .28886916 \cdot G' + .43601035 \cdot B'$$

$$V = .61497538 \cdot R' - .51496512 \cdot G' - .10001026 \cdot B'$$

To convert $Y \in [0, 1]$, $U \in [-.436, .436]$, $V \in [-.615, .615]$ to $R, G, B \in [0, 255]$:

$$R = 255 \cdot (Y + 1.13988 \cdot V)$$

$$G = 255 \cdot (Y - .39464 \cdot U - .58062 \cdot V)$$

$$B = 255 \cdot (Y + 2.03206 \cdot U)$$

APPENDIX D

EULER-LAGRANGE EQUATIONS FOR CHAN-VESE MODEL

We follow the notations in tutorial [165] and the hints[1] online, providing complete intermediate steps of the derivation.

$$\begin{aligned}\arg\min_{c_1,c_2,\varphi} E(c_1,c_2,\varphi) &= \mu\int_\Omega \delta(\varphi(x))\,|\nabla\varphi(x)|\,\mathrm{d}x + \nu\int_\Omega H(\varphi(x))\,\mathrm{d}x \\ &+ \lambda_1\int_\Omega |f(x)-c_1|^2 H(\varphi(x))\,\mathrm{d}x + \lambda_2\int_\Omega |f(x)-c_2|^2\,(1-H(\varphi(x)))\,\mathrm{d}x \end{aligned} \tag{D.1}$$

in which H denotes the Heaviside function and δ the Dirac mass:

$$H(t) = \begin{cases} 1 & t \geq 0, \\ 0 & t < 0, \end{cases} \qquad \delta(t) = \frac{\mathrm{d}}{\mathrm{d}t}H(t) \tag{D.2}$$

f is the original gray-scale image. μ, ν, λ_1, and λ_2 are real parameters. c_1 and c_2 are constants determined for segmentation. $\varphi(x) \in [-1,1]$ is the level-set function, and the zero level-set $\varphi(x) = 0$ specifies the boundary between interior and exterior regions.

The Euler-Lagrange equations w.r.t φ can be derived with the following derivative:

$$\left.\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\right|_{\varepsilon=0} E(\varphi(x)+\varepsilon\eta(x)) \tag{D.3}$$

To be concise, we let $\varphi_\varepsilon(x) = \varphi(x) + \varepsilon\eta(x)$ as suggested by Luminita Vese, and derive the equation for each component of the energy functional (D.1).

$$\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\int_\Omega \delta(\varphi_\varepsilon(x))\,|\nabla\varphi_\varepsilon(x)|\,\mathrm{d}x = \int_\Omega \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\left[\delta(\varphi_\varepsilon(x))\,|\nabla\varphi_\varepsilon(x)|\right]\mathrm{d}x \tag{D.4}$$

[1] https://dsp.stackexchange.com/questions/28941

$$= \int_\Omega \left[\frac{\mathrm{d}}{\mathrm{d}\varepsilon} \delta\left(\varphi_\varepsilon(x)\right) \right] |\nabla \varphi_\varepsilon(x)| + \delta\left(\varphi_\varepsilon(x)\right) \left[\frac{\mathrm{d}}{\mathrm{d}\varepsilon} |\nabla \varphi_\varepsilon(x)| \right] \mathrm{d}x \tag{D.5}$$

$$= \int_\Omega \delta'\left(\varphi_\varepsilon(x)\right) \eta(x) |\nabla \varphi_\varepsilon(x)| + \delta\left(\varphi_\varepsilon(x)\right) \left[\frac{\mathrm{d}}{\mathrm{d}\varepsilon} \left(\langle \nabla \varphi_\varepsilon(x), \nabla \varphi_\varepsilon(x) \rangle \right)^{\frac{1}{2}} \right] \mathrm{d}x \tag{D.6}$$

$$= \int_\Omega \delta'\left(\varphi_\varepsilon(x)\right) \eta(x) |\nabla \varphi_\varepsilon(x)| + \delta\left(\varphi_\varepsilon(x)\right) \left[\frac{\frac{\mathrm{d}}{\mathrm{d}\varepsilon} \langle \nabla \varphi_\varepsilon(x), \nabla \varphi_\varepsilon(x) \rangle}{2 |\nabla \varphi_\varepsilon(x)|} \right] \mathrm{d}x \tag{D.7}$$

$$= \int_\Omega \delta'\left(\varphi_\varepsilon(x)\right) \eta(x) |\nabla \varphi_\varepsilon(x)| + \delta\left(\varphi_\varepsilon(x)\right) \left[\frac{\frac{\mathrm{d}}{\mathrm{d}\varepsilon} \langle \nabla \left(\varphi(x) + \varepsilon \eta(x)\right), \nabla \left(\varphi(x) + \varepsilon \eta(x)\right) \rangle}{2 |\nabla \varphi_\varepsilon(x)|} \right] \mathrm{d}x \tag{D.8}$$

$$= \int_\Omega \delta'\left(\varphi_\varepsilon(x)\right) \eta(x) |\nabla \varphi_\varepsilon(x)| + \delta\left(\varphi_\varepsilon(x)\right) \left[\frac{\langle \nabla \varphi(x), \nabla \eta(x) \rangle + \varepsilon \langle \nabla \eta(x), \nabla \eta(x) \rangle}{|\nabla \varphi(x)|} \right] \mathrm{d}x \tag{D.9}$$

We now apply the boundary condition $\varepsilon = 0$ and eliminate those terms that contain ε.

$$\begin{aligned} &\left. \frac{\mathrm{d}}{\mathrm{d}\varepsilon} \right|_{\varepsilon=0} \int_\Omega \delta\left(\varphi(x)\right) |\nabla \varphi(x)| \, \mathrm{d}x \\ &= \int_\Omega \delta'\left(\varphi(x)\right) \eta(x) |\nabla \varphi(x)| + \delta\left(\varphi(x)\right) \left[\frac{\langle \nabla \varphi(x), \nabla \eta(x) \rangle}{|\nabla \varphi(x)|} \right] \mathrm{d}x \qquad \text{(D.10)} \\ &= \int_\Omega \delta'\left(\varphi(x)\right) \eta(x) |\nabla \varphi(x)| + \left[\left\langle \frac{\delta\left(\varphi(x)\right) \nabla \varphi(x)}{|\nabla \varphi(x)|}, \nabla \eta(x) \right\rangle \right] \mathrm{d}x \qquad \text{(D.11)} \end{aligned}$$

To further simplify the second term in the integral, we revisit Green's first identity:

$$\int_\Omega \left(\psi \Delta \phi + \nabla \psi \cdot \nabla \phi \right) \mathrm{d}x = \oint_{\partial \Omega} \psi (\nabla \phi \cdot \vec{n}) \, \mathrm{d}s \tag{D.12}$$

in which $\vec{n}$ is the outward normal on the image boundary.

In our case, we plug the following to (D.12).

$$\psi = \eta(x), \qquad \nabla \phi = \frac{\delta\left(\varphi(x)\right) \nabla \varphi(x)}{|\nabla \varphi(x)|} \tag{D.13}$$

Notice that $\Delta \phi = \operatorname{div} \nabla \phi$, we can rearrange (D.12) as follows.

$$\int_\Omega \left[\left\langle \frac{\delta\left(\varphi(x)\right) \nabla \varphi(x)}{|\nabla \varphi(x)|}, \nabla \eta(x) \right\rangle \right] \mathrm{d}x = \int_\Omega \nabla \psi \cdot \nabla \phi \, \mathrm{d}x = \oint_{\partial \Omega} \psi (\nabla \phi \cdot \vec{n}) \, \mathrm{d}s - \int_\Omega \psi \operatorname{div} \nabla \phi \, \mathrm{d}x \tag{D.14}$$

$$= \oint_{\partial\Omega} \eta(x) \left(\frac{\delta(\varphi(x))\,\nabla\varphi(x)}{|\nabla\varphi(x)|} \cdot \vec{n} \right) \mathrm{d}s - \int_{\Omega} \eta(x) \operatorname{div}\left(\frac{\delta(\varphi(x))\,\nabla\varphi(x)}{|\nabla\varphi(x)|} \right) \mathrm{d}x \tag{D.15}$$

We then apply the product rule of divergence $\operatorname{div}(g \cdot \vec{v}) = \nabla g \cdot \vec{v} + g \cdot \operatorname{div} \vec{v}$ to last term of (D.15). In this case, we have $g = \delta(\varphi(x))$ and $\vec{v} = \frac{\nabla\varphi(x)}{|\nabla\varphi(x)|}$.

$$\begin{aligned}
&- \int_{\Omega} \eta(x) \operatorname{div}\left(\frac{\delta(\varphi(x))\,\nabla\varphi(x)}{|\nabla\varphi(x)|} \right) \mathrm{d}x \\
&= - \int_{\Omega} \eta(x) \left[\nabla\delta(\varphi(x)) \cdot \frac{\nabla\varphi(x)}{|\nabla\varphi(x)|} + \delta(\varphi(x)) \cdot \operatorname{div}\left(\frac{\nabla\varphi(x)}{|\nabla\varphi(x)|} \right) \right] \mathrm{d}x && \text{(D.16)} \\
&= - \int_{\Omega} \eta(x) \left[\delta'(\varphi(x))\,\nabla(\varphi(x)) \cdot \frac{\nabla\varphi(x)}{|\nabla\varphi(x)|} + \delta(\varphi(x)) \cdot \operatorname{div}\left(\frac{\nabla\varphi(x)}{|\nabla\varphi(x)|} \right) \right] \mathrm{d}x && \text{(D.17)} \\
&= \int_{\Omega} -\delta'(\varphi(x))\,\eta(x)|\nabla\varphi(x)| - \delta(\varphi(x)) \operatorname{div}\left(\frac{\nabla\varphi(x)}{|\nabla\varphi(x)|} \right) \eta(x)\,\mathrm{d}x && \text{(D.18)}
\end{aligned}$$

We are now ready to plug (D.18) in (D.15), and then to (D.11).

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\bigg|_{\varepsilon=0} \int_{\Omega} \delta(\varphi_\varepsilon(x))\,|\nabla\varphi_\varepsilon(x)|\,\mathrm{d}x = \int_{\Omega} \delta'(\varphi(x))\,\eta(x)|\nabla\varphi(x)| + \left[\left\langle \frac{\delta(\varphi(x))\,\nabla\varphi(x)}{|\nabla\varphi(x)|}, \nabla\eta(x) \right\rangle \right] \mathrm{d}x \\
&= \int_{\Omega} \delta'(\varphi(x))\,\eta(x)|\nabla\varphi(x)|\,\mathrm{d}x + \oint_{\partial\Omega} \eta(x) \left(\frac{\delta(\varphi(x))\,\nabla\varphi(x)}{|\nabla\varphi(x)|} \cdot \vec{n} \right) \mathrm{d}s \\
&\quad - \int_{\Omega} \delta'(\varphi(x))\,\eta(x)|\nabla\varphi(x)|\,\mathrm{d}x - \int_{\Omega} \delta(\varphi(x)) \operatorname{div}\left(\frac{\nabla\varphi(x)}{|\nabla\varphi(x)|} \right) \eta(x)\,\mathrm{d}x && \text{(D.19)} \\
&= \oint_{\partial\Omega} \eta(x) \left(\frac{\delta(\varphi(x))\,\nabla\varphi(x)}{|\nabla\varphi(x)|} \cdot \vec{n} \right) \mathrm{d}s - \int_{\Omega} \delta(\varphi(x)) \operatorname{div}\left(\frac{\nabla\varphi(x)}{|\nabla\varphi(x)|} \right) \eta(x)\,\mathrm{d}x && \text{(D.20)}
\end{aligned}$$

For the rest three terms in (D.1), the derivations are much simpler than the first one. We let again that $\varphi_\varepsilon(x) = \varphi(x) + \varepsilon\eta(x)$.

For the length term, we have

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}\varepsilon} \int_{\Omega} H(\varphi_\varepsilon(x))\,\mathrm{d}x = \int_{\Omega} \frac{\mathrm{d}}{\mathrm{d}\varepsilon} H(\varphi_\varepsilon(x))\,\mathrm{d}x = \int_{\Omega} H'(\varphi_\varepsilon(x))\,\eta(x)\,\mathrm{d}x \\
&= \int_{\Omega} \delta(\varphi_\varepsilon(x))\,\eta(x)\,\mathrm{d}x = \int_{\Omega} \delta(\varphi(x) + \varepsilon\eta(x))\,\eta(x)\,\mathrm{d}x && \text{(D.21)}
\end{aligned}$$

We apply the boundary condition $\varepsilon = 0$ and obtain:

$$\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0} \int_\Omega H\left(\varphi_\varepsilon(x)\right) \mathrm{d}x = \int_\Omega \delta\left(\varphi(x)\right) \eta(x) \, \mathrm{d}x \tag{D.22}$$

Similarly, for the interior data term, we have:

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}\varepsilon} \int_\Omega |f - c_1|^2 H\left(\varphi_\varepsilon(x)\right) \mathrm{d}x = \int_\Omega \frac{\mathrm{d}}{\mathrm{d}\varepsilon} |f - c_1|^2 H\left(\varphi_\varepsilon(x)\right) \mathrm{d}x \\ &= \int_\Omega (f - c_1)^2 \delta\left(\varphi(x) + \varepsilon\eta(x)\right) \eta(x) \, \mathrm{d}x \end{aligned} \tag{D.23}$$

Applying the boundary condition $\varepsilon = 0$ leads to:

$$\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0} \int_\Omega |f - c_1|^2 H\left(\varphi_\varepsilon(x)\right) \mathrm{d}x = \int_\Omega (f - c_1)^2 \delta\left(\varphi(x)\right) \eta(x) \, \mathrm{d}x \tag{D.24}$$

Finally, for the exterior data term, we have:

$$\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0} \int_\Omega |f - c_2|^2 \left(1 - H\left(\varphi_\varepsilon(x)\right)\right) \mathrm{d}x = -\int_\Omega (f - c_2)^2 \delta\left(\varphi(x)\right) \eta(x) \, \mathrm{d}x \tag{D.25}$$

With results available for all terms in (D.20), (D.22), (D.24), and (D.25), we compute the directional derivative with the parameters μ, ν, λ_1 and λ_2.

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0} E\left(\varphi(x) + \varepsilon\eta(x)\right) \\ &= \mu \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0} \int_\Omega \delta\left(\varphi_\varepsilon(x)\right) \left|\nabla\varphi_\varepsilon(x)\right| \mathrm{d}x + \nu \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0} \int_\Omega H\left(\varphi_\varepsilon(x)\right) \mathrm{d}x \\ &\quad + \lambda_1 \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0} \int_\Omega |f - c_1|^2 H\left(\varphi_\varepsilon(x)\right) \mathrm{d}x + \lambda_2 \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0} \int_\Omega |f - c_2|^2 \left(1 - H\left(\varphi_\varepsilon(x)\right)\right) \mathrm{d}x \end{aligned} \tag{D.26}$$

$$\begin{aligned} &= \mu \oint_{\partial\Omega} \eta(x) \left(\frac{\delta\left(\varphi(x)\right) \nabla\varphi(x)}{|\nabla\varphi(x)|} \cdot \vec{n} \right) \mathrm{d}s - \mu \int_\Omega \delta\left(\varphi(x)\right) \operatorname{div}\left(\frac{\nabla\varphi(x)}{|\nabla\varphi(x)|} \right) \eta(x) \, \mathrm{d}x + \nu \int_\Omega \delta\left(\varphi(x)\right) \eta(x) \, \mathrm{d}x \\ &\quad + \lambda_1 \int_\Omega (f - c_1)^2 \delta\left(\varphi(x)\right) \eta(x) \, \mathrm{d}x - \lambda_2 \int_\Omega (f - c_2)^2 \delta\left(\varphi(x)\right) \eta(x) \, \mathrm{d}x \end{aligned} \tag{D.27}$$

$$= \mu \oint_{\partial\Omega} \eta(x) \left(\frac{\delta\left(\varphi(x)\right) \nabla\varphi(x)}{|\nabla\varphi(x)|} \cdot \vec{n} \right) \mathrm{d}s$$

$$- \int_{\Omega} \eta(x)\delta\left(\varphi(x)\right)\left[\mu \operatorname{div}\left(\frac{\nabla\varphi(x)}{|\nabla\varphi(x)|}\right) - \nu - \lambda_1(f-c_1)^2 + \lambda_2(f-c_2)^2\right]\mathrm{d}x \tag{D.28}$$

$$= \mu\left\langle \eta(x), \frac{\delta\left(\varphi(x)\right)\nabla\varphi(x)}{|\nabla\varphi(x)|}\cdot\vec{n}\right\rangle - \left\langle \eta(x), \delta\left(\varphi(x)\right)\left[\mu \operatorname{div}\left(\frac{\nabla\varphi(x)}{|\nabla\varphi(x)|}\right) - \nu - \lambda_1(f-c_1)^2 + \lambda_2(f-c_2)^2\right]\right\rangle \tag{D.29}$$

To ensure that the directional derivative vanishes for every direction, the following Euler-Lagrange equations must be satisfied:

$$\begin{cases} \dfrac{\delta\left(\varphi(x)\right)\nabla\varphi(x)}{|\nabla\varphi(x)|}\cdot\vec{n} = 0, & x \in \partial\Omega \\ \delta\left(\varphi(x)\right)\left[\mu \operatorname{div}\left(\dfrac{\nabla\varphi(x)}{|\nabla\varphi(x)|}\right) - \nu - \lambda_1(f-c_1)^2 + \lambda_2(f-c_2)^2\right] = 0, & x \in \Omega \end{cases} \tag{D.30}$$

Therefore, the gradient descent is the following.

$$\begin{cases} \dfrac{\delta\left(\varphi(x)\right)\nabla\varphi(x)}{|\nabla\varphi(x)|}\cdot\vec{n} = 0, & x \in \partial\Omega \\ \dfrac{\mathrm{d}\varphi}{\mathrm{d}t} = \delta\left(\varphi(x)\right)\left[\mu \operatorname{div}\left(\dfrac{\nabla\varphi(x)}{|\nabla\varphi(x)|}\right) - \nu - \lambda_1(f-c_1)^2 + \lambda_2(f-c_2)^2\right], & x \in \Omega \end{cases} \tag{D.31}$$

In practice, one should regularize the Heaviside function with one of the following options.

$$H_{1,\varepsilon} = \begin{cases} 1 & \text{if } z > \varepsilon \\ 0 & \text{if } z < -\varepsilon \\ \dfrac{1}{2}\left[1 + \dfrac{z}{\varepsilon} + \dfrac{1}{\pi}\sin\left(\dfrac{\pi z}{\varepsilon}\right)\right] & \text{otherwise} \end{cases}, \qquad H_{2,\varepsilon} = \frac{1}{2}\left(1 + \frac{2}{\pi}\arctan\left(\frac{z}{\varepsilon}\right)\right) \tag{D.32}$$

APPENDIX E

SUPPLEMENTARY MATERIAL:

VERIFYING THE CAUSES OF ADVERSARIAL EXAMPLES

E.1 Verification of the Statistical Explanations

E.1.1 Linearity of the classifier

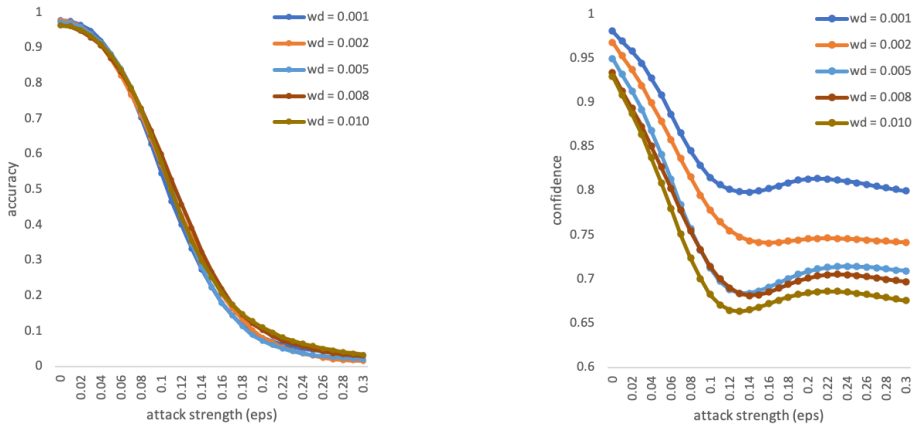
Hypothesis D *Adversarial perturbations can be magnified by the linear coefficients in the model, which further result in high prediction confidence for misclassification.*

Reasoning: According to [24], the output of $w^T x$ may differ substantially from that of $w^T(x + \Delta x)$, especially when the dimension is high.

Design principles: To weaken the linearity of classifiers, we can decrease the linear coefficients' scale via L_2 normalization implemented in weight decay: higher weight decays in the optimizer settings indicate larger L_2 penalties.

Experiment: We train fully-connected neural-network classifiers on MNIST [114] with varying weight decays in optimizer settings. The common network model consists of 4 layers as 784(relu)-200(relu)-200(relu)-10(softmax), with cross-entropy as the loss function. The classifiers are trained for 10 epochs with a batch size of 128 and Adam optimizer with a learning rate of 0.01. After obtaining the classifiers, we conduct FGSM with varying strength from and evaluate both the prediction accuracy and confidence on those generated adversarial test samples.

Figure E.1 plots both average accuracy and prediction confidence for classifiers trained with different weight decays. At the accuracy level (Figure E.1a), the curves frequently intersect with one another as the attack strength increases. Based on the accuracy curves only, we cannot conclude that the loss of accuracy on adversarial examples results directly from the linearity of classifiers. It is, however, safe to argue that model linearity is related



(a) accuracy (b) confidence

Figure E.1: Adversarial robustness of classifiers with different linearity: higher wd means stronger L_2 normalization and smaller absolute values in linear coefficients.

to the high prediction confidence according to the well-separated curves on average confidence (Figure E.1b). Moreover, the small bounce near $\epsilon = 0.14$ reflects the shift for most samples from a decrease of probability in the correct category to an increase of probability in the incorrect category.

E.1.2 Categories as events in a probability space

Hypothesis E *Classifiers tend to assign higher confidence at adversarial examples because all output probabilities must add up to 1.*

Reasoning: Similar arguments have been raised in [166] and [49]. Due to the lack of cushion classes for cases such as "do not know," the classifier has to pick the remaining category in high confidence once it rules out all that is impossible. Instead of learning the general features and activation, the classifier learns a posterior probability given the input must belong to one of the target categories.

Design principles: To break the constraint on one-sum probabilities, we replace the softmax activation for the final layer with sigmoid for each neuron, and substitute the cross-entropy (CE) loss with binary cross-entropy (BCE) loss. Consequently, each output probability satisfies its constraint on belonging to the range of $[0, 1]$, and the sum of

probabilities now ranges from 0 to the number of categories. Such a configuration is commonly used for training multi-label classifiers. For single-label classification problems, the category with the highest output probability is selected as the prediction.

Experiment: We train a pair of classifiers with the two combinations of activation and loss functions mentioned earlier, following the same configurations as described for verifying the previous hypothesis in Section E.1.1. Figure E.2 reveals interesting behaviors of the two classifiers under adversarial attacks in terms of classification accuracy and prediction confidence. One apparent observation is that the prediction confidence becomes lower when the one-sum constraint is lifted, which verifies hypothesis E. When the strength of the attack is weak ($\epsilon < .1$), accuracy decreases faster for the classifier trained using softmax and CE. When the attack becomes stronger ($\epsilon > .1$), however, accuracy from the one trained using sigmoid and BCE drops to a lower value. One may notice that the turning points for accuracy and confidence coincide somewhere around $\epsilon = .1$. Such a flipping point can be explained by the following phases[1] of attacking and the one-sum constraint.

Let us assume that the difficulty varies for adversarially perturbing a sample and alternating its predicted label. Accordingly, a particular attack that drives one sample to phase one may turn another sample to phase two. When attacks are weak during the first phase, their primary effect is to decrease the probability in the correct category until the output label is changed. Under the one-sum constraint, the probability decrease in one category would lead to an increase of probability in at least another category. On the contrary, the fall-and-rise bond is decoupled by sigmoid and BCE. Thus, the one-sum constraint from softmax and CE accelerates the phase of confidence decrease in the correct category. As attacks become stronger, they further push the prediction confidence on an incorrect category to climb higher at phase two. When the one-sum constraint is lifted by sigmoid and BCE, the prediction confidence for the incorrect class can increase more aggressively without considering other classes, and the accuracy may drop to near 0. In contrast, any further

[1] Although FGSM is a one-shot attack, we may treat a series of them with increasing ϵ as an evolution of a single attack for analysis.

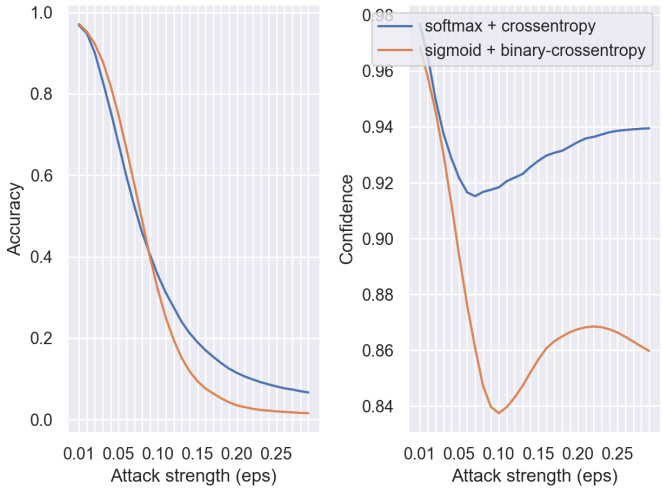


Figure E.2: Robustness comparison on classifiers trained with (i.e., softmax + crossentropy) and without (sigmoid + binary-crossentropy) the one-sum constraint on output probabilities.

confidence increase under the one-sum constraint with softmax and CE would require a decrease of probability at some other categories. Finally, the confidence fall-back for the classifier with sigmoid and BCE after $\epsilon = .2$ reflects that the decrease of probability from the hard-to-attack samples at phase one outweighs the increase of probability from those easy-to-attack samples at phase two.

E.1.3 Combination of linearity and one-sum probabilities

Hypothesis F *Adversarial examples result from the combination of two reasons: (1) linearity of the classifier, and (2) the one-sum requirement on output probabilities.*

Reasoning: Experiment results from the previous two hypotheses imply that linearity and one-sum constraint cause high prediction confidence on adversarial examples, but each of them alone may not be sufficient for explaining the significant loss of accuracy under adversarial attacks. This motivates us to consider whether the combination of the two is actually the game-changer.

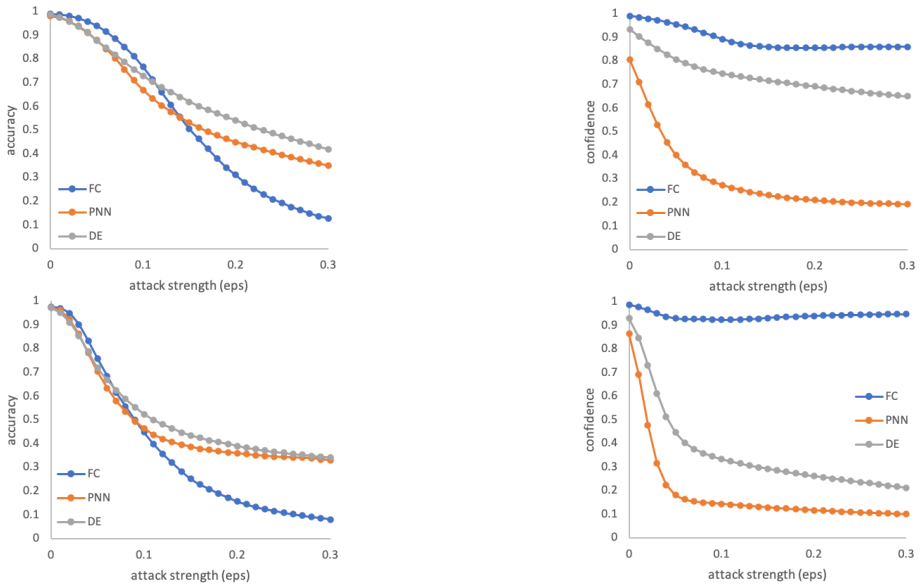
Design principles: To simultaneously remove the linear coefficient and break the one-sum constraint, we adopt MLP-PNN (denoted by PNN) and Density Estimator (DE), which

are proposed in [167]. Alternatively, one may follow a combined approach by introducing weight decay as in verifying Hypothesis D and performing the substitution as we did for Hypothesis E. The difference between those two strategies is that the PNN and DE completely remove the linear weights in the final layer (or equivalently setting to them as in unit scale 1), whereas weight decay penalizes large scale of coefficients at all layers.

Experiment: We can view the classifiers in two parts: the feature extractor and the head. Two options are available for the feature extractor part: we may either follow the same architecture with fully-connected layers like the one for verifying Hypothesis D or add two convolutional layers at the bottom. As for the network head, we experiment with three options: fully-connected layer (FC) with cross-entropy loss and the two proposed architectures (PNN and DE) with binary cross-entropy loss.

According to Figure E.3, the PNN and DE appear to be more adversarially robust than the commonly used MLP (bottom) or CNN (top), which are equipped with heads of fully-connected (FC) layers. Similar to the analysis for Figure E.2 in the previous hypothesis, the decrease of accuracy can be roughly divided into two phases (i.e., probability decrease of the correct class and probability increase of the incorrect class, for FC in particular). As the strength of attacks increases, the classification accuracy from PNN and DE gradually converge at a significantly higher level than FC; meanwhile, the prediction confidence keeps decreasing. In general, MLP and CNN should be considered as misleading because of their low classification accuracy and high prediction confidence. In contrast, the PNN and DE are at least one level more robust: being somewhat unreliable yet sensitive to perturbations in terms of confidence drop. Overall, results from Figure E.3 demonstrate that the combination of linearity and one-sum constraint brings a stronger impact to the existence of adversarial examples.

The comparison between PNN and DE reflects a trade-off from the flexibility of bias terms in the head layer. One major difference between PNN and DE is that the bias terms in the final layer are predefined and fixed in PNN but trainable in DE. As a consequence, DE



(a) accuracy (b) confidence

Figure E.3: Adversarial robustness for classifiers with different heads, tested with two types of feature extractor: convolutional (top row) and fully-connected layers (bottom row).

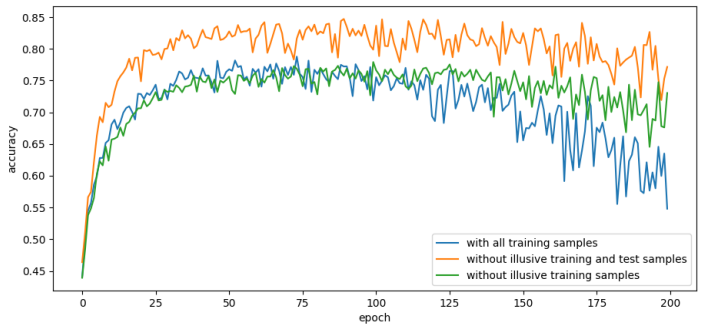


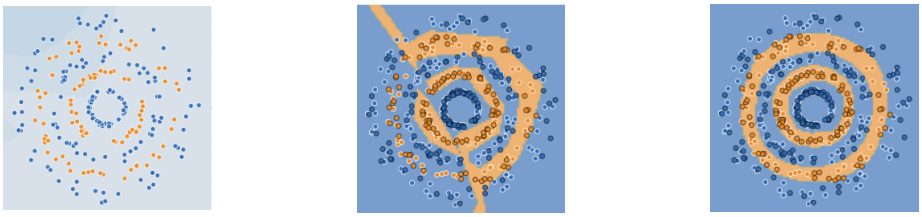
Figure E.4: Training without illusive samples help alleviate incorrect momentum.

achieves slightly higher accuracy on adversarial examples yet in much higher confidence. We will leave the interpretation and validation of such an observation as future work.

E.2 Alleviating Incorrect Momentum by Excluding Illusive Samples

It is observed that accuracy on both training and test samples may drop at late epochs because of the incorrect strong momentum[2] when training with optimizers such as RMSprop [168] and Adam [128]. Figure E.4 shows the accuracy curves for training with and without

[2]Reported in issue `https://github.com/keras-team/keras/issues/7603`



(a) training data (b) trained w/ (x_1, x_2) (c) trained w/ (x_1, x_2, x_1^2, x_2^2)

Figure E.5: Illustration on the uncertain bridges when classifying with insufficient capacity.

illusive samples: Despite a slight drop in test accuracy during early epochs, accuracy drop at late epochs is alleviated. If we further remove test samples that are consistently misclassified, the effect is weakened even more. A correlation seems to exist between the incorrect momentum and the illusive samples.

E.3 Uncertain Bridges from Limited Model Capacity

We illustrate the uncertain bridges created by a classifier with insufficient model capacity using a group of alternate concentric circles customized from Tnesorflow playground.[3] When the classifier is trained end-to-end on 2-tuples of (x_1, x_2), the network tries to connect training samples of the orange category by creating bridges between the two separated concentric circles. Under the constraints of limited capacity (i.e., 3-layer with size 8-4-2) and training budget (1,000 epochs), the classifier fails to connect those illusive samples because it is topologically impossible to connect all orange points without passing through the blue regions in 2-dimensional space[4]. When the model's capacity is increased by higher-order features x_1^2 and x_2^2, the classifier can easily solve the task under the same constraints (Figure E.5c).

[3] `http://playground.tensorflow.org`

[4] The toy example is designed for illustration purposed only. It is possible to classify the alternate concentric circles into separated regions with sufficient network capacity and densely populated training samples. Those two conditions, however, usually cannot be both satisfied for natural-image classification.

REFERENCES

[1] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[2] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[5] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *International Conference on Learning Representations*, 2018.

[6] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," in *International Conference on Learning Representations*, 2018.

[7] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.

[8] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, IEEE, 2011, pp. 1521–1528.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. B. Estrach, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[10] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, vol. 24, Elsevier, 1989, pp. 109–165.

[11] F. P. Schuller, *Lecture 2: Topological manifolds*, https://youtu.be/93f-ayezCqE?t=245, The WE-Heraeus International Winter School on Gravity and Light, 2015.

[12] J. M. Lee, "Smooth manifolds," in *Introduction to Smooth Manifolds*, Springer, 2003, pp. 1–29.

[13] X. He, W.-Y. Ma, and H.-J. Zhang, "Learning an image manifold for retrieval," in *Proceedings of the 12th annual ACM international conference on Multimedia*, ACM, 2004, pp. 17–23.

[14] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *European Conference on Computer Vision*, Springer, 2016, pp. 597–613.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[16] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick, "Neuroscience-inspired artificial intelligence," *Neuron*, vol. 95, no. 2, pp. 245–258, 2017.

[17] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba, "Undoing the damage of dataset bias," in *European Conference on Computer Vision*, Springer, 2012, pp. 158–171.

[18] T. Tommasi and T. Tuytelaars, "A testbed for cross-dataset analysis," in *European Conference on Computer Vision*, Springer, 2014, pp. 18–31.

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[20] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[21] K. Vodrahalli, K. Li, and J. Malik, "Are all training examples created equal? an empirical study," *arXiv preprint arXiv:1811.12569*, 2018.

[22] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *International Conference on Machine Learning*, 2017, pp. 1885–1894.

[23] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *International Conference on Machine Learning*, 2018, pp. 4334–4343.

[24] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[25] J. Gomes, *Adversarial attacks and defences for convolutional neural networks*, https://medium.com/onfido-tech/adversarial-attacks-and-defences-for-convolutional-neural-networks-66915ece52e7, 2018.

[26] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1765–1773.

[27] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[28] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, IEEE, 2016, pp. 372–387.

[29] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[30] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017, pp. 39–57.

[31] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[32] A. Mustafa, S. H. Khan, M. Hayat, J. Shen, and L. Shao, "Image super-resolution as a defense against adversarial attacks," *IEEE Transactions on Image Processing*, vol. 29, pp. 1711–1724, 2019.

[33] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. D. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *6th International Conference on Learning Representations, ICLR 2018*, 2018.

[34] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[35] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," *arXiv preprint arXiv:1711.00117*, 2017.

[36] R. R. Wiyatno, A. Xu, O. Dia, and A. de Berker, "Adversarial examples in modern machine learning: A review," *arXiv preprint arXiv:1911.05268*, 2019.

[37] X. Huang, D. Kroening, M. Kwiatkowska, W. Ruan, Y. Sun, E. Thamo, M. Wu, and X. Yi, "Safety and trustworthiness of deep neural networks: A survey," *arXiv preprint arXiv:1812.08342*, 2018.

[38] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2018.

[39] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, and B. Edwards, "Adversarial robustness toolbox v1.1.0," *CoRR*, vol. 1807.01069, 2018.

[40] A Goodfellow and B Papernot, "Is attacking machine learning easier than defending it?" *cleverhans-blog*, vol. 15, 2017.

[41] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, 2019.

[42] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," in *International Conference on Learning Representations*, 2018.

[43] C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 501–509.

[44] A. Fawzi, S.-M. Moosavi-Dezfooli, P. Frossard, and S. Soatto, "Empirical study of the topology and geometry of deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3762–3770.

[45] D. Mickisch, F. Assion, F. Greßner, W. Günther, and M. Motta, "Understanding the decision boundary of deep neural networks: An empirical study," *arXiv preprint arXiv:2002.01810*, 2020.

[46] J. Gilmer, N. Ford, N. Carlini, and E. Cubuk, "Adversarial examples are a natural consequence of test error in noise," in *International Conference on Machine Learning*, 2019, pp. 2280–2289.

[47] T. Tanay and L. Griffin, "A boundary tilting persepective on the phenomenon of adversarial examples," *arXiv preprint arXiv:1608.07690*, 2016.

[48] A. Shamir, I. Safran, E. Ronen, and O. Dunkelman, "A simple explanation for the existence of adversarial examples with small hamming distance," *arXiv preprint arXiv:1901.10861*, 2019.

[49] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein, "Are adversarial examples inevitable?" In *International Conference on Learning Representations*, 2018.

[50] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in *Advances in Neural Information Processing Systems*, 2019, pp. 125–136.

[51] L. Y. Pratt, "Discriminability-based transfer between neural networks," in *Advances in neural information processing systems*, 1993, pp. 204–211.

[52] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[53] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.

[54] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[55] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *International Conference on Learning Representations*, 2017.

[56] B. Peng, X. Jin, J. Liu, D. Li, Y. Wu, Y. Liu, S. Zhou, and Z. Zhang, "Correlation congruence for knowledge distillation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5007–5016.

[57] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1365–1374.

[58] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, and Z. Dai, "Variational information distillation for knowledge transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9163–9171.

[59] R Caruana, D. Silver, J Baxter, T. Mitchell, L. Pratt, and S Thrun, "Learning to learn: Knowledge consolidation and transfer in inductive systems," in *Workshop held at NIPS-95, Vail, CO*, 1995.

[60] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, "A kernel method for the two-sample-problem," in *Advances in neural information processing systems*, 2007, pp. 513–520.

[61] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*, PMLR, 2015, pp. 97–105.

[62] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[63] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, 2020.

[64] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" In *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[65] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International conference on machine learning*, 2014, pp. 647–655.

[66] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[67] S. Kornblith, J. Shlens, and Q. V. Le, "Do better imagenet models transfer better?" In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2661–2671.

[68] K. He, R. Girshick, and P. Dollár, "Rethinking imagenet pre-training," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4918–4927.

[69] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," *arXiv preprint arXiv:1904.07734*, 2019.

[70] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, and Z. Kira, "Re-evaluating continual learning scenarios: A categorization and case for strong baselines," in *NeurIPS Continual learning Workshop*, 2018.

[71] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[72] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, 2019.

[73] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*, Springer, 2016, pp. 694–711.

[74] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky, "Texture networks: Feed-forward synthesis of textures and stylized images.," in *ICML*, 2016, pp. 1349–1357.

[75] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," *Proc. of ICLR*, 2017.

[76] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua, "Stylebank: An explicit representation for neural image style transfer," in *Proc. CVPR*, vol. 1, 2017, p. 4.

[77] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[78] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[79] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8789–8797.

[80] Y. Hu, H. He, C. Xu, B. Wang, and S. Lin, "Exposure: A white-box photo post-processing framework," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 2, p. 26, 2018.

[81] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz, "A closed-form solution to photorealistic image stylization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 453–468.

[82] J. Yoo, Y. Uh, S. Chun, B. Kang, and J.-W. Ha, "Photorealistic style transfer via wavelet transforms," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9036–9045.

[83] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[84] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, and F. Durand, "Deep bilateral learning for real-time image enhancement," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 118, 2017.

[85] N. Murray, L. Marchesotti, and F. Perronnin, "Ava: A large-scale database for aesthetic visual analysis," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 2408–2415.

[86] S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann, and H. Winnemoeller, "Recognizing image style," in *Proceedings of the British Machine Vision Conference. BMVA Press*, 2014.

[87] V. Bychkovsky, S. Paris, E. Chan, and F. Durand, "Learning photographic global tonal adjustment with a database of input / output image pairs," in *The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[88] D. Mishkin, *A quick evaluation of different color-space preprocessing performance on imagenet-2012*, `https://github.com/ducha-aiki/caffenet-benchmark/blob/master/Colorspace.md`.

[89] S. N. Gowda and C. Yuan, "Colornet: Investigating the importance of color spaces for image classification," in *Asian Conference on Computer Vision*, Springer, 2018, pp. 581–596.

[90] S. Nowozin, B. Cseke, and R. Tomioka, "F-gan: Training generative neural samplers using variational divergence minimization," in *Advances in Neural Information Processing Systems*, 2016, pp. 271–279.

[91] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, 2017, pp. 214–223.

[92] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2794–2802.

[93] D. Berthelot, T. Schumm, and L. Metz, "Began: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[94] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are gans created equal? a large-scale study," in *Advances in neural information processing systems*, 2018, pp. 698–707.

[95] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017, pp. 5767–5777.

[96] A. Cauchy, "Méthode générale pour la résolution des systemes d'équations simultanées," *Comp. Rend. Sci. Paris*, vol. 25, no. 1847, pp. 536–538, 1847.

[97] R. Tyleček and R. Šára, "Spatial pattern templates for recognition of objects with regular structure," in *German Conference on Pattern Recognition*, Springer, 2013, pp. 364–374.

[98] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robustness of deep neural networks via stability training," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4480–4488.

[99] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *International Conference on Learning Representations*, 2017.

[100] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[101] E. L. Gettier, "Is justified true belief knowledge?" *analysis*, vol. 23, no. 6, pp. 121–123, 1963.

[102] B. C. Csáji, "Approximation with artificial neural networks," *Faculty of Sciences, Etvs Lornd University, Hungary*, vol. 24, p. 48, 2001.

[103] Wikipedia contributors, *Epistemology — Wikipedia, the free encyclopedia*, `https://en.wikipedia.org/w/index.php?title=Epistemology&oldid=932632279`, [Online; accessed 14-January-2020], 2019.

[104] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[105] K. Matsuoka, "Noise injection into inputs in back-propagation learning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 3, pp. 436–440, 1992.

[106] C. Wang and J. C. Principe, "Training neural networks with additive noise in the desired signal," *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1511–1517, 1999.

[107] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," Tech. Rep., 2013. arXiv: `1306.5151 [cs-cv]`.

[108] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, IEEE, 2008, pp. 722–729.

[109] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-UCSD Birds 200," California Institute of Technology, Tech. Rep. CNS-TR-2010-001, 2010.

[110] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 413–420.

[111] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[112] J. Lazarow, L. Jin, and Z. Tu, "Introspective neural networks for generative modeling," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2774–2783.

[113] *Tiny imagenet visual recognition challenge*, `https://tiny-imagenet.herokuapp.com/`, 2017.

[114] Y. LeCun, *The mnist database of handwritten digits*, `http://yann.lecun.com/exdb/mnist/`, 1998.

[115] P. Micaelli and A. J. Storkey, "Zero-shot knowledge transfer via adversarial belief matching," in *Advances in Neural Information Processing Systems*, 2019, pp. 9547–9557.

[116] H. Yin, P. Molchanov, J. M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz, "Dreaming to distill: Data-free knowledge transfer via deepinversion," in *The IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020.

[117] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[118] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[119] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images.," in *Iccv*, vol. 98, 1998, p. 2.

[120] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand, "A gentle introduction to bilateral filtering and its applications," in *ACM SIGGRAPH 2007 courses*, ACM, 2007, p. 1.

[121] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, IEEE, vol. 2, 2005, pp. 60–65.

[122] A. Yezzi, "Modified curvature motion for image smoothing and enhancement.," *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*, vol. 7, no. 3, pp. 345–352, 1998.

[123] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," in *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017.

[124] U. Jang, X. Wu, and S. Jha, "Objective metrics and gradient descent algorithms for adversarial examples in machine learning," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACM, 2017, pp. 262–277.

[125] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song, "Natural adversarial examples," *arXiv preprint arXiv:1907.07174*, 2019.

[126] T. Cai, J. Fan, and T. Jiang, "Distributions of angles in random packing on spheres," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 1837–1864, 2013.

[127] A. Ford and A. Roberts, "Colour space conversions," *Westminster University, London*, vol. 1998, pp. 1–31, 1998.

[128] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[129] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[130] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[131] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[132] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[133] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *BMVC*, 2016.

[134] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[135] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[136] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[137] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[138] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[139] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[140] U. Ozbulak, *Pytorch cnn visualizations*, `https://github.com/utkuozbulak/pytorch-cnn-visualizations`, 2019.

[141] E. Tseng, F. Yu, Y. Yang, F. Mannan, K. S. Arnaud, D. Nowrouzezahrai, J.-F. Lalonde, and F. Heide, "Hyperparameter optimization in black-box image processing using differentiable proxies," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–14, 2019.

[142] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[143] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep photo style transfer," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6997–7005.

[144] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[145] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 633–641.

[146] M Hossain, "Whitening and coloring transforms for multivariate gaussian random variables," *Project Rhea*, 2016.

[147] M. Perrot, N. Courty, R. Flamary, and A. Habrard, "Mapping estimation for discrete optimal transport," in *Advances in Neural Information Processing Systems*, 2016, pp. 4197–4205.

[148] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Transactions on image processing*, vol. 10, no. 2, pp. 266–277, 2001.

[149] P. Marquez-Neila, L. Baumela, and L. Alvarez, "A morphological approach to curvature-based evolution of curves and surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 1, pp. 2–17, 2013.

[150] A. Levin, D. Lischinski, and Y. Weiss, "A closed-form solution to natural image matting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 228–242, 2007.

[151] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[152] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, no. 3/4, pp. 321–377, 1936.

[153] J. A. Wegelin *et al.*, "A survey of partial least squares (pls) methods, with emphasis on the two-block case," *University of Washington, Tech. Rep*, 2000.

[154] M. Borga, "Canonical correlation: A tutorial," *On line tutorial http://people. imt. liu. se/magnus/cca*, vol. 4, no. 5, 2001.

[155] T. De Bie, N. Cristianini, and R. Rosipal, "Eigenproblems in pattern recognition," in *Handbook of Geometric Computing*, Springer, 2005, pp. 129–167.

[156] E. R. Cook, K. R. Briffa, and P. D. Jones, "Spatial regression methods in dendroclimatology: A review and comparison of two techniques," *International Journal of Climatology*, vol. 14, no. 4, pp. 379–402, 1994.

[157] H. R. Glahn, "Canonical correlation and its relationship to discriminant analysis and multiple regression," *Journal of the atmospheric sciences*, vol. 25, no. 1, pp. 23–31, 1968.

[158] L. Sun, S. Ji, S. Yu, and J. Ye, "On the equivalence between canonical correlation analysis and orthonormalized partial least squares," in *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[159] B. M. Wise, *Properties of partial least squares (pls) regression, and differences between algorithms*, http://www.eigenvector.com/Docs/Wise_pls_properties.pdf.

[160] Quality and Technology, *Partial least squares regression 1 introduction (3/4)*, https://www.youtube.com/watch?v=5VtDAzjMAUA&t=476s, 2011.

[161] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[162] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "Mulan: A java library for multi-label learning," *Journal of Machine Learning Research*, vol. 12, pp. 2411–2414, 2011.

[163] E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, and I. Vlahavas, "Multi-target regression via input space expansion: Treating targets as inputs," *Machine Learning*, vol. 104, no. 1, pp. 55–98, 2016.

[164] S. Džeroski, D. Demšar, and J. Grbović, "Predicting chemical parameters of river water quality from bioindicator data," *Applied Intelligence*, vol. 13, no. 1, pp. 7–17, 2000.

[165] P. Getreuer, "Chan-vese segmentation," *Image Processing On Line*, vol. 2, pp. 214–224, 2012.

[166] Y. Fan and A. Yezzi, "Towards an understanding of neural networks in natural-image spaces," *arXiv preprint arXiv:1801.09097*, 2018.

[167] H. Li, P. Barnaghi, S. Enshaeifar, and F. Ganz, "Continual learning using task conditional neural networks," *arXiv preprint arXiv:2005.05080*, 2020.

[168] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.